Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Institut für Theoretische und Technische Informatik
Fachgebiet Rechnerarchitekturen



Diplomarbeit

Konzeption und Realisierung eines fairen Software-Kopierschutzes basierend auf einer Client/Server-Architektur

Diplomarbeit zur Erlangung des akademischen Grades Diplominformatiker vorgelegt der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau von:

Michael Kunze

Matr.-Nr.: 25148

Verantw. Hochschullehrer:	Prof. Dr.-Ing. habil. Wolfgang Fengler
Hochschulbetreuer:	Dr.-Ing. Jürgen Nützel
Datum:	Jena, 1. September 2004
Inventarisierungsnummer:	2004-08-02/063/IN96/2231

Vorwort

An dieser Stelle möchte ich mich bei all den Menschen bedanken, die mir bei der Erstellung dieser Arbeit behilflich waren. Mein Betreuer, Herr Dr.-Ing. Jürgen Nützel, der mir jeder Zeit mit Rat und Tat zur Seite stand und mit Anregungen und Verbesserungsvorschlägen wesentlich zur Qualität dieser Diplomarbeit beigetragen hat; die Mitarbeiter Stefan Richter und Marko Langbein der 4FriendsOnly.com AG, die mir meine zahlreichen Fragen bezüglich der Game-Feature-Plattform geduldig beantwortet haben; Mario Voigt, ..., die sich die Mühe gemacht haben und die Arbeit Korrektur zu lesen; Frau H. Biewald, die mir bei der Kontrolle der Rechtschreibung geholfen hat; Torsten Mangner, der mich bei der Erstellung der Grafiken unterstützte.

Mein wohl größter Dank geht an meine Eltern, Eckhard und Uta Kunze. Ihr habt mir dieses Studium erst ermöglicht. Ich möchte ich mich für das in mich gesetzte Vertrauen und die finanzielle Unterstützung bedanken.

Vielen Dank.

Selbstständigkeitserklärung

Ich versichere an Eides statt durch eigenhändige Unterschrift, die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der aufgeführten Quellen und Hilfsmittel verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, habe ich als solche kenntlich gemacht. Ich weiß, dass bei Abgabe einer falschen Versicherung die Prüfung als nicht bestanden zu gelten hat. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Jena, 1. September 2004

Michael Kunze

Thesen dieser Arbeit

- Das Urheberrecht wurde bereits im Jahre 1993 erweitert, um Software vor Diebstahl zu schützen.
- Software verlangt nach zusätzlichem Schutz vor illegalen Kopien.
- Shareware ist ein einfaches Vertriebskonzept zur preiswerten Verteilung von Software.
- Die „Next-Generation Secure Computing Base“-Initiative von Microsoft ist der erste Versuch einer sicheren Computerplattform für Windows auf Betriebssystemebene.
- Die meisten Kopierschutzmechanismen für Softwareprodukte ignorieren den Fairness-Gedanken.
- Die 4FriendsOnly.com Internet Technologies AG hat die Game-Feature-Plattform (GFP) entwickelt.
- Die bestehende Game-Feature-Plattform ist ein monolitischer Block. Sie ist zu unflexibel und daher verbesserungswürdig.
- Der Web-Service ist die ideale Technologie für die Verteilung von Softwarekomponenten.
- Durch einen Web-Service ist es möglich, den Archiv-Builder der GFP flexibel in andere Systeme einzubinden.
- Der Shop-Betreiber hat die Möglichkeit, seine Vorstellungen des „Fair-Use“-Gedankens durch eigene Regeln mit Hilfe des Archiv-Builders der GFP umzusetzen.

Jena, 1. September 2004

Michael Kunze

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Software-Unternehmen verlieren nach eigenen Angaben jährlich bis zu zwölf Milliarden Dollar durch Raubkopien, die Musikindustrie beziffert ihre Verluste mit 4,6 Milliarden Dollar weltweit, die Filmindustrie geht von 3,5 Milliarden an Verlusten aus. Dies stellt einen geschätzten Gesamtverlust von 20 Millarden US Dollar dar, was in etwa einem Prozent des Deutschen Bruttoinlandproduktes entspricht – Tendenz steigend.

Schon lange werden neben mehr Problembewusstsein auch strengere gesetzliche Maßnahmen sowie mehr Fahndungsdruck von Seiten der Exekutive gefordert. Teilweise bereits in dem erneuerten Urheberrecht vom September 2003 umgesetzt, fehlt hier jedoch noch die flächendeckende Durchsetzung. Die Privatkopie stellt auch in Zukunft eine rechtliche Grauzone dar.

Microsoft – Initiator der *Business Software Alliance (BSA)* – geht hier eigene Wege. Mit Einführung von Office XP wurde der Benutzer erstmalig zur so genannten Produktaktivierung gezwungen. 30 Tage nach Installation wird eine Freischaltung nötig oder das Programm stellt seine Arbeit ein. Mittels des Produktschlüssels und eines Telefons hat der User einen Code bei Microsoft zu erfragen, der ihn zur endgültigen Benutzung des Programmes berechtigt. Die Alternative ist die Deinstallation. Ziel soll es sein, Raubkopien und deren Verteilung einzuschränken.

Auf der einen Seite haben wir die Industrie, die ihre Investitionen durch Schutzmaßnahmen sichern möchte und dies auch, wie wir sehen werden, mittels verschärfter Gesetze durchzudrücken versucht. Auf der anderen Seite steht der Ottonormalverbraucher. Auch er möchte seine Interessen gewahrt sehen und fair behandelt werden. Aus irgend einem Grund werden Schikanen bei einem neuen Softwareprodukt leichter akzeptiert, als dies bei einem Autokauf der Fall sein würde.

Die Mitarbeiter der 4FriendsOnly.com Internet Technologies AG haben sich ihre eigenen Gedanken zum Thema Kopierschutz gemacht und ein System entwickelt, das versucht, die

oben angesprochenen Probleme zu lösen. Wie sich aber herausstellte, gibt es noch einige Schwächen, die es zu beseitigen gilt.

1.2 Ziel der Arbeit

Ziel der Arbeit soll es sein, die Game-Feature-Plattform (kurz: GFP) der 4FriendsOnly.com AG auf Schwächen zu untersuchen und zu analysieren. Dazu sollen zunächst grundlegende Technologien anderer Firmen untersucht werden. Aufbauend auf die GFP sollen dann die Probleme auf der Server-Seite beseitigt und dies mittels einer Referenzapplikation belegt werden.

1.3 Aufbau der Arbeit – Kapitelübersicht

Diese Arbeit ist so aufgebaut, dass man sie von vorn nach hinten durchlesen oder zu einzelnen Kapiteln springen kann.

In Kapitel 2 wird der aktuelle Stand der Technik näher untersucht, dazu gehören bereits existierende Lösungen, Technologien sowie Produkte auf dem Markt. Zuerst erfolgt ein kleiner Einstieg in das Urheberrecht, um dem geneigten Leser die Thematik des Kopierschutzes und dessen Nutzen näher zu bringen. Im Anschluss erfolgt ein kurzer Abriss darüber, wie heutige (Produktaktivierung von Microsoft) und zukünftige (Next-Generation Secure Computing Base) Technologien Software schützen sollen.

Für die Game-Feature-Plattform notwendige Kenntnisse über kryptografische Algorithmen werden in Kapitel 3 vermittelt. Hierbei handelt es sich lediglich um einen groben Überblick, da man mit diesem Thema ein ganzes Buch füllen könnte.

In Kapitel 4 wird der Leser etwas über die Game-Feature-Plattform der Firma 4FriendsOnly.com AG erfahren. Im Anschluss daran soll ein Konzept zur Umsetzung eines fairen Softwarekopierschutzes auf Basis einer Client/Server-Architektur vorgestellt werden. Im Kapitel 5 wird eine Basisimplementierung vorgestellt.

Eine Zusammenfassung sowie ein Ausblick auf zukünftige Entwicklungen wird die Arbeit abrunden.

Kapitel 2

Grundlagen und Voraussetzungen

Im nun folgenden Kapitel soll ein Ausgangspunkt für die vorliegende Arbeit geschaffen werden. Ziel ist es, Technologien oder Verfahren bzw. Produkte vorzustellen, die zu diesem Zeitpunkt verfügbar oder gerade in der Entstehung begriffen sind. Dies soll als Grundlage für das darauf folgende Kapitel dienen und die Schwierigkeiten sowie Probleme aufzeigen.

2.1 Das Urheberrecht

Schutz für Kreative und Investitionen – oder Diebstahl von Allgemeingut? Über die Frage, was das Urheberrecht ist, was es bewirken und wovor es schützen soll, scheiden sich die Geister. Wie viel Schutz für das geistige Eigentum brauchen wir wirklich?

Schon das Wort löst bei manchen heftige Reaktionen aus, lässt die Emotionen kochen. Das ist auch kein Wunder, denn es geht um Geld und Freiheit. Großes wird da heraufbeschworen: „Urheberrecht schafft die Grundlage für die totale Überwachung der Netze!“, rufen die einen. „Investitionen müssen durch starke Rechte gesichert werden“, halten die anderen entgegen.

„Das Urheberrecht schützt den Urheber in seinen geistigen und persönlichen Beziehungen zum Werk und in der Nutzung des Werkes“ -– sagt jedenfalls das Urheberrechtsgesetz (Vgl. §11 UrhG). Das Bundesverfassungsgericht geht mit der Aussage *„Urheberrecht ist Eigentum“* sogar noch einen Schritt weiter und definiert damit gleichzeitig die zweite Dimension des Urheberrechts, die im Grundgesetz verankert ist: *„Eigentum verpflichtet“* (Vgl. Art. 14, Abs. 2 GG). Damit ist im Grunde alles klar: Das Urheberrecht soll einen Ausgleich zwischen den Interessen der Allgemeinheit an der möglichst ungehinderten Nutzung wertvoller Inhalte und den Interessen der Rechtsinhaber an Kontrolle und wirtschaftlicher Verwertung schaffen. Das deutsche Urheberrecht behält dem Berechtigten vor, über jede Art von Verwertung zu entscheiden und hieran wirtschaftlich teilzuhaben – anders gesagt, steht dem Schöpfer eines Werkes das alleinige Verwertungsrecht zu. Um das Wesen des Urheberrechtes besser zu verstehen soll nun ein kurzer geschichtlicher Exkurs folgen.

2.1.1 Hintergrund und Geschichte

Die Rechte des Urhebers, wie wir sie heute kennen, waren den antiken Völkern noch völlig fremd. Sie konnten mit der Idee des geistigen Eigentums nichts anfangen. Schriftsteller, Maler, Bildhauer und Architekten wurden wie Handwerker behandelt. Sie wurden entlohnt und hatten darüber hinaus keine weiteren Ansprüche an den von ihnen geschaffenen Werken.

Bis in das späte Mittelalter wurden Bücher einzeln hergestellt, verkauft und bezahlt. Als erfolgreich galt, wer über die Mauern von Schlössern und Burgen bekannt war und dessen Werke Aufnahme in die entstehenden Bibliotheken der neu gegründeten Universitäten fand. Als Mitte des 15. Jahrhunderts der Buchdruck erfunden wurde, war es möglich, Bücher in größeren Mengen und für einen anonymen Markt zu produzieren. War um 1400 die Bibliothek des französischen Königs Karl V mit gerade einmal 1.000 handgeschriebenen Bänden in ganz Europa berühmt, gab es 150 Jahre später in den meisten Städten schon Staats- und Landesbibliotheken mit zehntausenden von gedruckten Büchern.

Um den gestiegenen Bedarf aller Bibliotheken zu decken, begann man Bücher nachzudrucken. Zum ersten Mal in der Geschichte reifte die Erkenntnis, dass die Freiheit des Nachdrucks den ursprünglichen Verlegern und Schriftstellern keine Einnahmen mehr erbrachte. Schon Martin Luther, bekanntlich ein sehr erfolgreicher Schriftsteller, beklagte sich heftig. In seiner *„Vorrhede oder Vermahnunge an die Drucker"* aus dem Jahre 1525 beklagte er sich wortreich über den unerlaubten Nachdruck und die Verfälschung seiner Werke:
„Es ist ein ungleich Ding, daß wir arbeiten und Kost sollen drauf wenden, und andere sollen genießen ... ."

So entschied man sich, Nachdruckverbote für einzelne Werke zu erlassen. Bereits im ausgehenden 15. Jahrhundert war es in Venedig verboten, Bücher bis fünf Jahre nach ihrem Erscheinen nachzudrucken. Im deutschen Reich, damals noch heilig, kamen im 16. Jahrhundert die kaiserlichen, später auch landesherrlichen und reichsstädtischen Druckprivilegien auf. Wer dem zuwiderhandelte, musste mit einer empfindlichen Geldstrafe und der Beschlagnahmung der Raubkopien rechnen. Diese Privilegien dienten aber in erster Linie dem Schutz der Verleger und der Sicherung des Absatzes als dem Schutz geistigen Eigentums. So konnte man kaum von einem Urheberschutz sprechen. Eher wurden Gewerbemonopole und Zunftzwänge auf das neue Medium „Buch" ausgeweitet.

In der Renaissance rückte die Individualität wieder mehr in den Vordergrund. Maler begannen Ihre Bilder zu signieren und Kopien der Bilder wurden von den Künstlern selber in deren Werkstätten erstellt. Schließlich gewährte man sogenannte „Autorenprivilegien", mit denen der Schöpfer für sein Werk belohnt wurde. Albrecht Dürer erhielt in Deutschland ein solches Recht und so manchem berühmten Maler wurden durch den jeweiligen Landesherrn Sonderprivilegien eingeräumt. Dieser Schutz bezog sich jedoch nur auf die Person (Persönlichkeitsrecht) nicht auf sein Werk. Auch wurde es willkürlich verliehen ohne einklagbaren Rechtsanspruch und brachte den Urhebern somit keine Einnahmen. Das geschriebene Werk

wurde auch weiterhin als Sache betrachtet. Mitte des 16. Jahrhunderts wurden Territorialprivilegien eingeführt, die allgemeine Nachdruckverbote in einem bestimmten Gebiet für einen begrenzten Zeitraum darstellten.

Erst im 18. Jahrhundert wurde die Theorie vom geistigen Eigentum entwickelt. In einem englischen Gesetz von 1710 wurde als erstes ein ausschließliches Vervielfältigungsrecht des Autors anerkannt (intellectual property). Dieses Recht traten die Autoren dann an die Verleger ab. Nach Ablauf der vereinbarten Zeit fielen alle Rechte wieder an den Autor zurück. Das Werk musste im Register der Buchhändlergilde eingetragen werden und erhielt einen sogenannten „Copyright“-Vermerk. So war es geschützt. Zum ersten Mal wurde ein unauflösbares Verhältnis zwischen Werk und Schöpfer konstatiert, das nicht verkauft oder aufgelöst werden konnte – nur die Nutzung war handelbar.

In den neu gegründeten Vereinigten Staaten von Amerika wurde das geistige Eigentum unter den Schutz der Verfassung gestellt. Kurz darauf wurde auch in Frankreich durch zwei Revolutionsgesetze von 1791 und 1793 das „propriété littéraire et artistique“[1] anerkannt. Preußen folgte am 11. Juni 1837 mit dem „Gesetz zum Schutze des Eigenthums an Werken der Wissenschaft und Kunst in Nachdruck und Nachbildung“. Es war das ausführlichste und zugleich modernste Urheberrechtsgesetz seiner Zeit. Noch im selben Jahr beschloss die Bundesversammlung (Deutscher Bund) eine 10-jährige Schutzfrist seit Erscheinen eines Werkes, die 1845 auf 30 Jahre – beginnend mit dem Tode des Urhebers – verlängert wurde (Vgl. [VGW04]).

Urheberrechtlichen Schutz kann ein Staat nur in seinem Hoheitsbereich gewährleisten. Mit der weltweiten Verbreitung kam daher das Bedürfnis nach weitergehendem Schutz auf. Zuerst wurden zu diesem Zwecke einzelne zwischenstaatliche Vereinbarungen getroffen. 1886 kam es zu einem Zusammenschluss von zehn Staaten, darunter Deutschland, in der Berner Übereinkunft. In der Folgezeit kam es zu mehreren Revisionen; seit 1908 spricht man daher von der „Revidierten Berner Übereinkunft“ (RBÜ). Die RBÜ sichert den Angehörigen eines Verbandsstaats einen Mindestschutz zu sowie die Gleichbehandlung unter dessen Angehörigen.

Das Urheberrecht in Deutschland trat vor nunmehr 40 Jahren – am 9. September 1965 – in Kraft. Doch die Technik entwickelte sich ständig weiter. Computer wurden immer kleiner und waren für jedermann erschwinglich. Mit dem Beginn des digitalen Zeitalters wurde auch das Erstellen von Kopien immer leichter. Längst vorbei sind die Zeiten, wo Leute mit einem Kassettenrekorder vor dem Radio saßen, um Lieder aufzunehmen. Heute „greift“ man den Mpeg2-Stream der D-Box ab, um seine Lieblingssendung zu einem späteren Zeitpunkt in der selben Qualität zu sehen oder brennt eine CD mit Beethovens Symphonien, um diese zu Hause im besten Raumklang zu geniessen. Eine digitale Kopie ist von ihrem Original nicht mehr zu unterscheiden.

Um diesem Trend entgegen zu wirken, wurde 1974 die World Intellectual Property Or-

[1] propriété littéraire et artistique, frz: geistiges Eigentum

ganization[2] gegründet. Die WIPO entstand als Teilorganisation der UNO, um die Rechte an immateriellen Gütern weltweit zu fördern. Heute gehören ihr 177 Mitgliedsstaaten an und sie überwacht 21 internationale Abkommen. Auch dort erkannte man die Popularität von Internet-Tauschbörsen, bei denen täglich millionenfache Copyright-Verletzungen stattfinden. So entschied man sich im Dezember 1996. zwei Abkommen zu verabschieden: das WIPO Copyright Treaty und das WIPO Performances and Phonograms Treaty.

2.1.2 Digital Millennium Copyright Act (DMCA)

Die USA waren nun gezwungen, gesetzgeberische Maßnahmen zu treffen, um die beiden WIPO-Abkommen zu erfüllen. Das neue Gesetz, welches am 28. Oktober 1998 einstimmig durch den US-Senat angenommen wurde, trug den Namen Digital Millenium Copyright Act. Es ist ein umstrittenes Gesetz, welches die Rechte von Copyright-Inhabern erweitert. Dieser Act soll die neu entstanden Verhältnisse regeln, die sich aus der Möglichkeit ergeben, durch digitale Reproduktion perfekte Kopien zu erstellen. Der Digital Millenium Copyright Act besteht aus fünf Teilen.

- Title I, **„WIPO Copyright and Performances and Phonograms Treaties Implementation Act“**, implementiert die beiden WIPO-Abkommen. Er erweitert das amerikanische Copyright-Gesetz um das Verbot der Umgehung von Kopierschutzmechanismen sowie das Verbot des Veränderns von *Copyright Management Information (CMI)*. Zusätzlich wird von den Herstellern von analogen Videorekordern gefordert, dass diese mit einem Kopierschutzsystem versehen sind.

- Title II, **„Online Copyright Infringement Liability Limitation Act“**, schafft einen sicheren Hafen für Internet-Service-Provider (ISP). Er gewährt dem Provider Straffreiheit, wenn er das Material, was zur Copyright-Verletzung geführt hat, sofort nach Bekanntwerden entfernt. Außerdem verpflichtet er den Internet-Service-Provider zu einer Sicherstellung der Beweise für eine spätere Sichtung durch die zuständigen Behörden.

- Title III, **„Computer Maintenance Competition Assurance Act“**, modifiziert das Copyright-Gesetz so, dass es Leuten, die Rechner reparieren, erlaubt, zeitweilige Kopien der Software zu erstellen, während sie an den Geräten arbeiten.

- Title IV, **„Miscellaneous Provisions“**, enthält eine ganze Reihe von Regelungen. So zum Beispiel wurden die Rechte und Pflichten des Copyright Office näher erläutert. Rundfunkanstalten wird es gestattet, temporäre Kopien zu erstellen. Des weiteren wurden Vorschriften hinzugefügt, die Bibliotheken das Aufbewahren von Kopien von Tonaufnahmen erlauben.

[2] World Intellectual Property Organization, engl: Weltorganisation für Geistiges Eigentum

- Title V, **„Vessel Hull Design Protection Act“**, fügt Abschnitte 1301 bis 1332 hinzu, um Designs von Schiffsrümpfen zu schützen.

Die Leute, die den DMCA verabschiedet haben, behaupten, dass er nötig war, um die Richtlinien der WIPO in amerikanisches Recht umzusetzen. Allerdings waren auch genau diese Lobbyisten für die Erstellung des Abkommens verantwortlich – eine sehr merkwürdige Praxis. Andere behaupten, dass bereits vor der Verabschiedung des DMCA alle nötigen Gesetze geschaffen waren, um das Recht auf Landesebene umzusetzen. Somit ist und bleibt der Digital Millennium Copyright Act ein umstrittenes Gesetz. Es gibt bereits Bestrebungen im Senat, das Gesetz zu modifizieren. Richard Boucher – ein demokratischer Kongressabgeordneter aus Virginia – gilt als einer der Anführer dieser Bewegung durch die Vorstellung des Digital Media Consumers’ Rights Act (kurz: DMCRA).

Die Erfahrungen in den USA mit dem DMCA haben unter anderem gezeigt, dass allein ein simpler Tastendruck auf einer Computertastatur bzw. die Veröffentlichung der „Anleitung zur Umgehung eines Kopierschutzes“ eine Anklage nach sich ziehen kann. Der DMCA scheint auch eine bedeutende Rolle in Auseinandersetzungen zu spielen. So hat Google aufgrund von Behauptungen von Scientology, eine Seite eines Scientologykritikers verletze ihre Rechte gemäß DMCA, prompt diese Seite kommentarlos und ungeprüft aus ihrem Index entfernt. Kritiker sprachen von Zensur.

Viele Firmen benutzen das DMCA als erweitertes Patent ohne Möglichkeit auf Lizenzen und ohne zeitliche Beschränkung. Hier einige Beispiele:

- Lexmark hat im Dezember 2002 gegen Static Control Components geklagt, da sie Kartuschen hergestellt haben, die in Lexmark Druckern funktionierten. Dazu musste SCC einen Chip in die Kartusche einbauen, der sich genauso wie bei Originalkartuschen verhielt. Lexmark hat gewonnen. Epson stellt seit längerem ebenfalls Kartuschen mit Chips her. Drittanbieter liefern deshalb keine Kartuschen für neuere Epson Drucker mehr aus.

- XBox Mod-Chips werden auf Basis des DMCA verfolgt und verboten. Diese Modifikationen werden meist benutzt, um Linux zu installieren und die Möglichkeiten der XBox zu erweitern. Das fällt eigentlich auch unter die Ausnahme des Reverse Engineerings.

- Die DVD Copy Control Association hat mit Berufung auf den DMCA am 17.Januar 2000 eine einstweilige Verfügung gegen Webseiten erwirkt, die DeCSS-Programme anboten oder Weblinks darauf setzten. DeCSS wurde entwickelt, um CSS-geschützte DVDs auf Linux und anderen Betriebssystemen abzuspielen. Im DCMA ist so etwas als eine Ausnahme erwähnt, nämlich zum Zweck des Reverse Engineerings. Das Gerichtsverfahren gegen den Entwickler von DeCSS, Jon Lech Johansen, endete mit einem Freispruch.

- Im Januar 2003 hat die Chamberlain Group gegen Skylink geklagt, weil sie einen kompatiblen Garagentüröffner in Form einer Fernbedienung herstellten und hat am 29. August 2003 verloren. Obwohl der Kopierschutz von Chamberlain gebrochen wurde, hat der Garagentürbesitzer das Recht, in seine Garage zu gelangen, selbst wenn er die Fernbedienung verloren hat.

- Für Aufsehen sorgte auch der Prozess gegen Dmitry Skylarov. Adobe hatte ihn wegen Verstosses gegen den DMCA angezeigt, da er auf US-Boden vor Publikum zeigen wollte, dass Adobe die ROT-13 „Verschlüsselung“ als „Kopierschutz“ für ihre E-Books verwendete und wie leicht diese „entschlüsselt“ werden kann. Er verbrachte 6 Monate (Juli bis Dezember 2001) im Gefängnis in den USA. Im Dezember 2003 endete sein Prozeß ebenfalls mit Freispruch (Vgl. [WIK04a]).

2.1.3 EUCD und die Umsetzung in deutsches Recht

Auch die europäische Union hatte sich durch Unterzeichnung der WIPO-Abkommen zur Handlung verpflichtet. So verabschiedete man am 22. Mai 2001 die Richtlinie 2001/29/EG (Vgl. [EUCD01]). Die als European Union Copyright Directive bekannt gewordene Direktive stellt das Gegenstück zum amerikanischen Digital Millennium Copyright Act dar. Sie setzt die von der EU unterzeichneten WIPO-Abkommen auf europäischer Ebene um.

Als EU-Mitglied machte Deutschland am 11. April 2003 sein Gesetz EUCD-konform, indem sie die so genannte Urheberrechts-Novelle verabschiedete. Das neue Urheberrechtsgesetz trat 5 Monate später, am 13. September 2003 in Kraft.

„Wirksame technische Maßnahmen zum Schutz eines nach diesem Gesetz geschützten Werkes oder eines anderen nach diesem Gesetz geschützten Schutzgegenstandes dürfen ohne Zustimmung des Rechtsinhabers nicht umgangen werden, soweit dem Handelnden bekannt ist oder den Umständen nach bekannt sein muss, dass die Umgehung erfolgt, um den Zugang zu einem solchen Werk oder Schutzgegenstand oder deren Nutzung zu ermöglichen.“[URHG]

So lautet Absatz 1 des neu ins Urheberrechtsgesetz aufgenommenen Paragraphen 95a, der den Abgeordneten anscheinend so gut gefiel, dass sie den Entwurf gänzlich unverändert im Bundeskabinett passieren ließen. Damit wurde auch erstmals in Deutschland die Umgehung von Kopierschutzmaßnahmen rechtlich sanktioniert.

Das mag Informatikern absurd erscheinen, für Juristen aber sind die Zeilen ein gefundenes Fressen. Denn der zweiten Absatz des neuen Paragraphens erklärt so ziemlich jede technische Schutzbemühung der Industrie für „wirksam“. Dort heißt es sybillinisch im schönsten Bürokratendeutsch:

„[...] Technische Maßnahmen sind wirksam, soweit durch sie die Nutzung eines geschützten Werkes oder eines anderen nach diesem Gesetz geschützten Schutzgegenstandes von dem Rechtsinhaber durch eine Zugangskontrolle, einen Schutzmechanismus wie Verschlüsselung, Verzerrung oder sonstige Umwandlung oder einen Mechanismus zur Kontrolle der

Vervielfältigung, die die Erreichung des Schutzziels sicherstellen, unter Kontrolle gehalten wird.“[URHG]

Vor allem die Musikindustrie kann also erst einmal aufatmen. Die auf immer mehr CDs aufgebrachten Kopierschutzsperren müssen zwar laut §95d zukünftig „*deutlich sichtbar gekennzeichnet werden*“, doch dafür sitzen diese mit Hilfe des Gesetzgebers dann felsenfest.

Dem Nutzer zu Hause drohen zwar keine strafrechtlichen Folgen, sollte er trotz des klaren Umgehungsverbots heimlich zu Clone-CD oder ähnlichen Programmen greifen, jedoch ist mit der Verschärfung des Gesetzes die Privatkopie faktisch abgeschafft. Dies ist vielen Verbraucherschutzverbänden und der Initiative zur Rettung der Privatkopie[3] ein Dorn im Auge. Bekommen Anwälte der Musik- oder Filmindustrie nämlich – etwa durch eine absichtlich oder unbeabsichtigt ins Internet gewanderte Kopie der Kopie – Wind von der häuslichen Umgehungsorgie, ist die zivilrechtliche Klage mitsamt Schadensersatzforderungen vorprogrammiert.

[3] http://www.privatkopie.net

2.2 Businessmodels für Software

Im nun folgenden Abschnitt sollen einige Vertriebskonzepte für Software dargelegt werden. Dazu soll dem Leser deutlich gemacht werden, welche Möglichkeiten es gibt, mit Software Geld zu verdienen. In diesem Zusammenhang wird auch häufig die Rede von „Computerprogrammen“ sein. Bevor nun ein Einstieg in die Problematik erfolgen soll, bedürfen die hier verwendeten Ausdrücke einer kurzen Erläuterung.

Die Bezeichnung „Programme für die Datenverarbeitung“ wurde 1985 in den Katalog der urheberrechtlich geschützten Werke gemäß § 2 Abs. 1 UrhG aufgenommen und fand somit eine Zuordnung zu den Sprachwerken. 1993 wurde dieser Terminus durch „Computerprogramme“ ersetzt. Jedoch schon vor 1985 wurde von einzelnen Kommissionen versucht, vornehmlich technisch orientierte Definitionen für den Begriff „Computerprogramm“ zu finden, welche aber teilweise divergierten. Entsprechend einer Mustervorschrift der *World Intellectual Property Organisation (WIPO)* ist ein Computerprogramm *„eine Folge von Befehlen, die nach Aufnahme in einen maschinenlesbaren Träger fähig sind, zu bewirken, daß eine Maschine mit informationsverarbeitenden Fähigkeiten eine bestimmte Funktion oder Aufgabe oder ein bestimmtes Ergebnis anzeigt, ausführt oder erzielt“*. Als entscheidendes Kriterium wurde dabei die Steuerungsfunktion eines Programms betont.

Mit der Einführung des § 69a Abs. 1 UrhG im Jahre 1993 erfuhr der Begriff eine Klarstellung und Erweiterung: Compuerprogramme sind danach Programme in jeder Gestalt, einschließlich des Entwurfmaterials. Zu einem Computerprogramm im urheberrechtlichen Sinne zählen somit bereits erste Lösungsskizzen für eine Programmentwicklung, das gesamte Entwurfs- und Ausführungsmaterial, wie Pflichtenhefte und Programmbeschreibungen, bis hin zum fertigen Programm. Hinzu kommt die Gesamtheit der Benutzerdokumentation, worunter insbesondere Anwenderhandbücher zu verstehen sind.

Anstelle von „Computerprogramm“ findet auch häufig die Bezeichnung „Software“ Verwendung. Bei Software handelt es sich um einen Sammelbegriff, dessen Umfang nicht ganz eindeutig ist. Versucht wird daher, Software durch eine negative Definition als alles zu beschreiben, was nicht Bestandteil der Hardware ist, wobei man unter „Hardware“ alle physikalischen Einheiten versteht, aus welchen sich eine Datenverarbeitungsanlage zusammensetzt.

Bei Shareware, Freeware und Public Domain handelt es sich um Computerprogramme, welche sich weder inhaltlich, noch funktional von anderer Software unterscheiden. Programme, die mit einer dieser Bezeichnungen versehen sind, werden vielmehr im Rahmen eines besonderen Vertriebskonzeptes verbreitet, welches eigenen schuldrechtlichen und urheberrechtlichen Regeln unterworfen ist. Hauptcharakteristikum dieser Programme ist, dass der Besitzer einer Programmkopie dazu ermutigt wird, selbständig eine Verbreitung dieser Software zu fördern, indem er Kopien anfertigt und an Dritte verteilt beziehungsweise Dritten das Kopieren ermöglicht. Die Verbreitung der Programme soll somit im Rahmen eines sogenannten Schneeballsystems erfolgen. Durch die exponentiell ansteigende Programmverviel-

fältigung soll ein hoher Verbreitungsgrad der jeweiligen Software auch ohne den Vertrieb durch ein Händlernetz erreicht werden. Programme, die auf diesem Wege vertrieben werden, werden gemeinhin in die drei Kategorien (Shareware, Freeware und Public Domain Software) untergliedert, wobei jeder dieser Bezeichnungen unterschiedlich weit gesteckte Befugnisse für den Benutzer des Programms signalisiert.

2.2.1 Public Domain Software

Ausgangspunkt der Entwicklung war die aus den USA stammende Public Domain Software. Bei dieser Form von Software handelte es sich ursprünglich um die Programme, welche mit Hilfe von amerikanischen Steuergeldern zu Zwecken der militärischen und zivilen Nutzung entwickelt wurden. Die entwickelten Programme wurden in der Folgezeit teilweise der breiten Öffentlichkeit zugänglich gemacht. Da die Finanzierung der Entwicklungskosten aus öffentlichen Mitteln erfolgte, wurde jedermann erlaubt, die Produkte frei zu benutzen und zu kopieren. Die Programme gehörten der *public domain* an, also zu den nicht durch ein Copyright geschützten Erzeugnissen. Jedem war somit erlaubt, mit dem Programm zu arbeiten, Veränderungen an ihm vorzunehmen und es zu verbreiten.

Als Mitte der 80er Jahre der Durchbruch des Personal Computers (PC) zu verzeichnen war, und er zunehmend auch seinen Platz in privaten Haushalten fand, stieg auch gleichzeitig die Zahl privater Programmierer. Die von diesem Personenkreis entwickelten Programme wurden der Öffentlichkeit zugänglich gemacht und daher als Public Domain Software deklariert. Eine Gegenleistung für ihre Programmiertätigkeit erwarteten sie nicht.

2.2.2 Freeware

Im Laufe der Zeit kristalisierte sich heraus, dass es Programmautoren gab, welche zwar ihre Programme jedermann zur Verfügung stellen wollten, gleichzeitig aber nicht wünschten, dass Veränderungen jedweder Art daran vorgenommen würden. Es bildete sich eine zweite Kategorie von Software, die so genannte Freeware.

Freeware (eigentlich ’freie Ware’) ist im allgemeinen Sprachgebrauch die übliche Bezeichnung für Software-Produkte, die ohne Vergütung zu bekommen sind und weitgehend ohne Einschränkung weiterverbreitet werden dürfen. Ein Autor kann generell für sein Werk, für das er per Gesetz ein Urheberrecht besitzt, die vertraglichen Bedingungen – damit sind die Rechte anderer an diesem Werk gemeint – in weitem Umfange festlegen. Ihm steht zu, diese frei zu geben oder eben nicht. Im Falle von Freeware heißt das zumeist, dass auf diese nach dem Urheberrecht bestehende Möglichkeit, die Verbreitung und Bearbeitung einschränken zu können, verzichtet wird. Außerdem wird auf eine sonst übliche Vergütung für die Nutzung von Kopien verzichtet. Es kann mehrere Gründe geben, ein Programm als Freeware zu veröffentlichen. Der Hauptgrund ist wohl, dass es sich bei manchen Programmen mangels Zielgruppe einfach nicht lohnt, für diese Programme Geld zu nehmen.

Freeware ist kein rechtsgültig definierter Begriff, da nicht vereinbart ist, was eigentlich genau frei ist. Daher ist in jedem Einzelfall anhand der Lizenzbedingungen zu prüfen, welche Rechte nun vorliegen.

GNU[4] ist zum Beispiel in gewissem Sinn Freeware, dennoch sind die Regelungen zur Nutzung und zur Weitergabe sehr genau und auf ganz spezielle Weise geregelt um die freie Weitergabe auch in Zukunft sicherzustellen.

Nach dem heutigen Verständnis wird die Bezeichnung „Public Domain Software“ für diejenigen Programme gebraucht, welche in nahezu beliebiger Weise verändert und auch in veränderter Form weitergegeben werden dürfen. Freeware darf hingegen nur unverändert weitergegeben werden.

2.2.3 Shareware

Zunächst einmal handelt es sich hier um Software wie jede andere. Vom kleinen Tool bis zur großen Anwendung kann man inzwischen fast alles als „Shareware“ bekommen.

„Stellen Sie sich vor, Sie möchten ins Kino gehen. Von dem angekündigten Film haben Sie vorher ein paar gute Meinungen gehört und die Plakate sind recht vielversprechend. An der Kasse werden Sie freundlich darauf hingewiesen, daß der Eintrittspreis erst am Ende der Vorstellung fällig ist... und auch nur dann, wenn Ihnen der Film gefallen hat. Klingt gut, nicht wahr? Genau nach diesem Prinzip funktioniert Shareware: Software, die Sie ausgiebig testen, bevor Sie sie bezahlen.“[HWF92]

Etwa zeitgleich – also Anfang der 80er Jahre – wurde das Sharewarekonzept entwickelt. Shareware teilt mit Freeware und Public Domain Software die Eigenschaft, dass sie von jedermann vervielfältigt und weitergegeben werden darf, ohne dass dadurch Urheberrechte verletzt werden. Voraussetzung: Dies geschieht kostenlos oder nur gegen Erhebung einer geringen Kopiergebühr, um Aufwandskosten für Diskette, Porto und ähnliches zu decken. Allerdings erwartet ein Sharewareautor für die dauerhafte Benutzung der Programme grundsätzlich eine Vergütung. Die Besonderheit des Sharewaresystems besteht darin, dass derjenige, welcher in den Besitz einer Programmkopie gelangt, das Programm zunächst einmal für eine bestimmte oder unbestimmte Dauer erproben darf. Shareware ist fair zum Kunden. Es wird ihm gestattet, ja es ist sogar erwünscht, die Software zu testen und somit herauszufinden, ob sie den eigenen Ansprüchen genügt. Shareware verlangt aber auch Fairness vom Kunden. Entschließt man sich, das Programm weiter zu benutzen, wird erwartet, dass ein gewisses Entgelt an den Autor entrichtet wird. Die Höhe dieses Entgelts und die Adresse, an welche der Betrag zu übersenden ist, sind in aller Regel einer Textdatei zu entnehmen, welche beim Start des Programms angezeigt wird.

Überweist der Anwender den Betrag nicht, so soll er zu einer Weiterbenutzung des Programms nicht mehr befugt sein; es bleibt ihm allerdings weiterhin gestattet, beziehungsweise

[4] http://www.gnu.org/

es ist sogar erwünscht, dass er das Programm weiter an Dritte verteilt. Unabhängig davon, ob der Anwender sich nun zur Nutzung des Programms entschließt oder nicht, darf er grundsätzlich keine Änderungen an dem Programm vornehmen.

Das Problem bei dieser Form der Softwareüberlassung zeigt sich darin, dass der Autor einer Shareware faktisch keine Kontrollmöglichkeiten darüber hat, ob jeder einzelne Anwender, der das Programm einer dauerhaften Verwendung zugeführt hat, auch den Betrag, welcher häufig als Registriergebühr oder Lizenzgebühr bezeichnet wird, an ihn entrichtet. Tatsächlich weiß der Programmautor in den seltensten Fällen, wie oft das entsprechende Programm überhaupt kopiert wurde und in wessen Händen sich eine Programmkopie befindet.

Grundsätzlich macht sich derjenige, welcher ohne die erforderlichen Nutzungsrechte ein Computerprogramm benutzt, einer Urheberrechtsverletzung schuldig. Er sieht sich daher nicht nur zivilrechtlichen Schadensersatzansprüchen und Ansprüchen nach §69f UrhG ausgesetzt, seine Handlung ist gleichzeitig strafrechtlich relevant. Es stellt sich daher die Frage, wann der Nutzer von Shareware die Schwelle zur Urheberrechtsverletzung überschreitet, denn die Besonderheit dieser Form von Programmnutzung besteht ja gerade darin, dass die Bestimmung dieser „Schwelle“ sowohl der Rechtsprechung, als auch der Literatur noch erhebliche Schwierigkeiten bereitet.

Dennoch soll nicht verschwiegen werden, dass es sehr unwahrscheinlich ist, für eine Urheberrechtsverletzung wegen unberechtigter Nutzung von Shareware zur Verantwortung gezogen zu werden. In der Realität bedeutet dies, dass der Sharewareautor mehr oder weniger vom guten Willen des Anwenders abhängig ist. Vielfach wird daher auch versucht, an Ehre oder Gewissen zu appellieren, um auf diesem Wege eine Zahlung zu bewirken. So finden sich häufig etwa dahingehende Ausführungen: „Seien Sie fair gegenüber den Autoren und honorieren Sie deren Arbeit und Idee“

Viele Sharewareautoren wollen nicht von der Redlichkeit ihrer Programmnutzer abhängig sein und versuchen daher, mit technischen Mitteln eine Zahlung herbeizuführen, indem sie die Möglichkeit der Programmnutzung in verschiedenster Art und Weise beschränken. Dies geschieht etwa dadurch, dass nicht alle Funktionen des Programms ausführbar sind, dass bei einem Computerspiel beispielsweise nicht alle Schwierigkeitsstufen zur Verfügung stehen oder aber der Programmlauf in immer kürzer werdenden Abständen unterbrochen wird, oft verbunden mit einer auf dem Bildschirm erscheinenden Aufforderung, nunmehr endlich die Registriergebühr zu entrichten.

Die Erfahrung zeigt allerdings, dass derartige Beschränkungen die Akzeptanz der Programme nicht erhöht haben, sondern vielmehr gegenteilige abschreckende Wirkung auf die Anwender ausgeübt haben, so dass man vielfach von dieser Praxis wieder abgewichen ist.

Mit der Registrierung verspricht der Sharewareautor weitere Vorteile: Soweit es sich um eine beschränkte Programmversion handelt, verpflichtet sich der Autor in erster Linie zur Lieferung einer Vollversion. Daneben werden aber zumeist noch weitere Leistungen ver-

sprochen, wie etwa die Zusendung einer Benutzerdokumentation in Form eines Handbuchs, die Versorgung mit weiteren Information, Unterstützung bei der Installation und Fehlersuche sowie Updates auf neuere Versionen.

Als Begründer der Sharewareidee wird gelegentlich der US-Amerikaner Jim Knopf genannt. Tatsächlich jedoch entstand die Sharewareidee gleichzeitig an zwei Orten um das Jahr 1982. Einmal in Tiburon, Kalifornien durch Andrew Fluegelman mit seinem Programm PC-Talk und in Bellevue, Washington durch Jim Knopf, der das Programm PC-File entwickelte. Es fing mit einem einfachen Programm zum Ausdrucken von Adressaufklebern an. Aus der Not heraus, damit auch mehr machen zu können, wurde ein Datenbankprogramm geboren, indem man alle mögliche Daten speichern konnte.

Da genau in dieser Zeit der Personal Computer seinen Siegeszug antrat, war die Nachfrage nach solchen Programmen groß. Jim Knopf verteilte sein selbstgeschriebenes Programm an Freunde und Arbeitskollegen. Diese wiederum begannen auch ihrerseits das Programm weiterzugeben. Bald war die Schar der Nutzer so groß, dass es finanziell nicht mehr haltbar war, alle per Post über Updates oder Neuerungen zu informieren. Aus diesem Grund fügte der Autor eine Meldung in das Programm ein, ihn bei der Entwicklung mit einer Spende von 10 Dollar zu unterstützen. Die Nachfrage war gewaltig. Einer der Nutzer machte Jim Knopf darauf aufmerksam, dass ein anderes Programm eine ähnliche Nachricht enthielte. So entschieden sich beide dazu, dieses ungewöhnliche Marketingexperiment zu verfolgen. Damit waren die ersten zwei Sharewareprogramme geboren. Zu ihrer Spitzenzeit verzeichnete die Firma von Jim Knopf über 4,5 Millionen Dollar Umsatz und beschäftigte 35 Mitarbeiter (Vgl. [KNO96]).

2.2.4 Weitere Formen

Es existieren noch andere Formen der Softwareüberlassung. Diese stellen aber in der Regel nur Sonderformen der drei genannten Hauptkategorien dar und sollen deshalb hier nur am Rande erwähnt werden:

- Als **Bookware** bezeichnet man Programme, die als Diskette oder CD-ROM einem Buch, zumeist einem Benutzerhandbuch, für eine bestimmte Software beigefügt sind. Bookware kann, muß aber nicht, einer der drei Kategorien zugeordnet werden, wobei es sich aber in den seltensten Fällen um Public Domain Software handeln wird.

- Unter **Bannerware** versteht man Programme, die Firmen aller Branchen zu Werbe- und Promotionszwecken auf den Markt bringen. Diese Programme enthalten dann oft Beschreibungen des beworbenen Produktes oder einfach nur Computerspiele, wobei das Markenlogo natürlich entsprechend in Szene gesetzt wird.

- Bei **Timeware** handelt es sich um Sharewareprogramme, welche nach Ablauf einer gewissen Zeitspanne nicht mehr benutzt werden können, ohne das Programm neu zu

starten. Dies soll eine Schikane des Benutzers darstellen, um ihn zu zwingen, die Vollversion zu ordern.

- Ähnlich verhält es sich mit **Registerware**. In gewissen Zeitabständen erscheint auf dem Bildschirm eine Nachricht, die den Anwender dazu auffordert, sich registrieren zu lassen. Während der Meldung wird der Programmablauf unterbrochen. Dabei werden die Zeitspannen zwischen den einzelnen Aufforderungen oft sukzessive verkürzt beziehungsweise die Dauer der Aufforderung verlängert.

- Eine eher seltene Form stellt die **Aidware** dar. Diese Shareware wird aus karitativen Beweggründen mit der Bitte versehen, die Registriergebühr, statt an den Sharewareautoren an eine gemeinnützige Organisation oder Einrichtung zu entrichten. Die Zahlung wird dann so behandelt, als ob sie an den Autor gegangen wäre.

- Bei der so genannten **Demoware** handelt es sich weder um Shareware noch um Freeware oder Public Domain Software. Vielmehr gibt Demoware ein Beispiel von dem Inhalt und Ablauf des fertigen Programmes, ohne das der Anwender Einfluß auf den Programmablauf hat. Der Benutzer bleibt auf die Rolle des Zuschauers beschränkt. Der einzige Zweck einer Demoware besteht darin, den potentiellen Anwender zum Kauf zu bewegen. Diese Form ist besonders bei Spielen sehr verbreitet, um die wartende Community zu ködern und weiter bei der „Stange“ zu halten.

- Als **Crippleware** bezeichnet man Programme, die eigentlich als Shareware konzipiert waren, aufgrund zahlreicher Beschränkungen oder Fehler beim Programmablauf aber dermaßen „verkrüppelt“ sind, dass eine sinnvolle Erprobung der Software nicht mehr möglich ist, da eine Vielzahl der Funktionen nicht oder nur zum Teil benutzbar sind, gilt diese Variante als Testversion gemeinhin als unbrauchbar.

In Abgrenzung zu den oben genannten Vertriebsformen soll der Terminus **Kaufsoftware** Verwendung finden. Dieser weder juristisch noch in Fachkreisen etablierte Begriff soll lediglich kennzeichnen, dass es sich hierbei um Standardsoftware handelt, die wie andere Waren in einem Geschäft oder online über das Internet erworben oder bestellt wird. Die Vervielfältigung ist hier weder erwünscht noch erlaubt und stellt somit das totale Gegenteil zu den bis jetzt behandelten Absatzformen dar. Mit dem Erwerb einer Lizenz ist man zu der Benutzung des Stück Softwares ermächtigt. Oftmals schließt das einen professionellen Support für einen gewissen Zeitraum ein.

2.3 Next-Generation Secure Computer Base

Wir haben bis jetzt das Thema nur von einer Seite beleuchtet – nämlich, dass es dem Konsumenten nicht erlaubt ist, einen vorhandenen Kopierschutz zu umgehen und somit – laut Gesetz – zum Urheberrecht gezwungen wird. Doch wer möchte die rund 20,5 Millionen[5] Computer in den privaten Haushalten kontrollieren? In der Bevölkerung fehlt nach wie vor das Verständnis für diese Thematik und Raubkopien werden immer noch als Bagatelle angesehen.

Es gilt also eine Lösung zu finden, die es dem Benutzer erst gar nicht erlaubt, gegen das Gesetz zu verstoßen und Raubkopien zu erstellen. Ein möglicher Ansatz wäre die direkte Integration in das Betriebssystem. Hier ist Microsoft – als wohl einer der größten Monopolisten auf diesem Gebiet – gefragt. Doch wie wir im Verlauf sehen werden, läßt auch dieser Ansatz einiges zu Wünschen übrig. Die größte Gefahr besteht wohl darin, dass ein Computerbenutzer die Herrschaft über seinen PC an Dritte abgibt.

„Next-Generation Secure Computer Base“ (kurz: NGSCB) ist die Bezeichnung von Microsoft für eine neue Initiative, die die Datensicherheit erhöhen und den Datenschutz gewährleisten soll. Früher bekannt als Palladium findet man folgende Definition auf der Homepage von Microsoft:

„Palladium is the code name for an evolutionary set of features for the Microsoft Windows operating system. When combined with a new breed of hardware and applications, these features will give individuals and groups of users greater data security, personal privacy, and system integrity. In addition, Palladium will offer enterprise customers significant new benefits for network security and content protection.“[MS02]

NGSCB ist keine Software, es ist kein Betriebssystem und auch keine Hardware. Vielmehr ist es ein Vorschlag für eine Architekturverbesserung eben dieser Komponenten. Erstmalig soll es in dem Nachfolger von Windows XP integriert werden. Das zukünftige Betriebssystem ist heute unter dem Namen „Longhorn“ bekannt. Da es bis zum heutigen Zeitpunkt noch nicht veröffentlicht wurde, weist Microsoft darauf hin, dass alle Informationen – NGSCB betreffend – noch nicht endgültig sind.

Das Problem vor dem die Softwareindustrie steht, ist schnell erklärt:

2.3.1 Das Problem

Trotz des Fortschritts heutiger Personalcomputer in Sachen Sicherheit und Privatsphäre wurde immer die Rückwärtskompatibilität mit alten Systemen gewahrt. Das Aufgeben dieser Strategie hätte vermutlich deutlich kleinere, schnellere und sicherere Systeme im Laufe der Jahre produziert. Jedoch ist die Industrie nach wie vor zum Verfolgen des eingeschlagenen Weges gezwungen, um Investitionen in Software, Hardware und Benutzertraining auch wei-

[5] laut Statistischem Bundesamt hatten 53% der 38,7 Millionen Haushalte im Jahr 2001 mindestens einen Personal Computer

terhin zu sichern. Ein Produkt, was alles bis dahin da gewesene über den Haufen wirft, umfangreiche Schulungen für Personal und langwierige Einarbeitungszeiten notwendig macht, hat wenig Chancen von den Konsumenten angenommen zu werden.

Allerdings hat die Entwicklung des verteilten Netzwerkes – im speziellen des Internets – neue Probleme und neuen Bedarf an vertrauenswürdiger Rechentechnik geschaffen. Dies wird auch durch die ständige Zunahme von privaten Informationen in einer digital vernetzten Welt bestätigt, was wiederum immer größere Herausforderungen mit sich bringt. Denn wer heute eine kritische Anwendung, zum Beispiel E-Banking, auf seinem Rechner ausführt und sensible Daten, beispielsweise Firmendokumente, verarbeitet, muss darauf vertrauen, das diese Anwendung nicht durch Schadprogramme (Viren, Trojaner oder Würmer) beeinflusst wird.

2.3.2 NGSCB als Lösung

Nun hat sich Microsoft vorgenommen, das Problem zu lösen und die Next-Generation Secure Computer Base Initiative der breiten Öffentlichkeit vorgestellt. NGSCB ist kein eigenständiges Betriebssystem, sondern eine Kombination aus Software- und Hardwarekomponenten, die in die nächste Generation von Betriebssystemen integriert sein soll. Microsoft propagiert vier Key Feature:

- **Strong process isolation** Das ist ein Mechanismus, der Daten und Programme im Speicher schützt. Hier kann nur zugreifen, wer im Zuge einer Überprüfung als vertrauenswürdig eingestuft wurde. Applikationen, die im „nexus mode“ laufen, können in diesem Bereich nicht durch andere Applikationen oder Betriebssysteme geändert, beobachtet bzw. abgehört werden.

- **Sealed storage** Äquivalent zum letzten Feature widmet sich dieses dem Schützen der Daten des Anwenders auf seiner Festplatte. Die geschriebenen Daten können nur durch das Programm gelesen werden, das sie verschlüsselt hat.

- **Secure paths** Dieses Feature soll die sichere Kommunikation zwischen Eingabegeräten und NCAs sowie Monitor – also Ausgabegeräte – und NCAs sicherstellen. Ziel ist es, dass Daten, die vom User eingegeben wurden und auf dem Bildschirm präsentiert werden, in keiner Weise von einem unberechtigten Programm (Trojaner) gelesen oder verändert werden können.

- **Attestation** Dieser Mechanismus wird für die Authentifizierung von Software und Hardware verantwortlich sein. Microsoft sieht hier eine lokale und remote Authentifizierung über das Internet vor.

Auf Grund der Tatsache, dass Longhorn noch nicht verfügbar ist, sind alle hier dargestellten Informationen mit Vorsicht zu genießen. Microsoft hat der Öffentlichkeit auf seiner

Homepage bereits einen ersten Blick auf die Funktionsweise von NGSCB erlaubt.

Das Betriebssystem Windows wird auch weiterhin der Vermittler zwischen Hardware und den Programmen sein, die auf dieser Architektur ausgeführt werden sollen. Mit der Einführung von NGSCB wird Windows in eine sogenannte „Left-Hand-Side“ und „Right-Hand-Side“ unterteilt (siehe *Abbildung 2.1*).

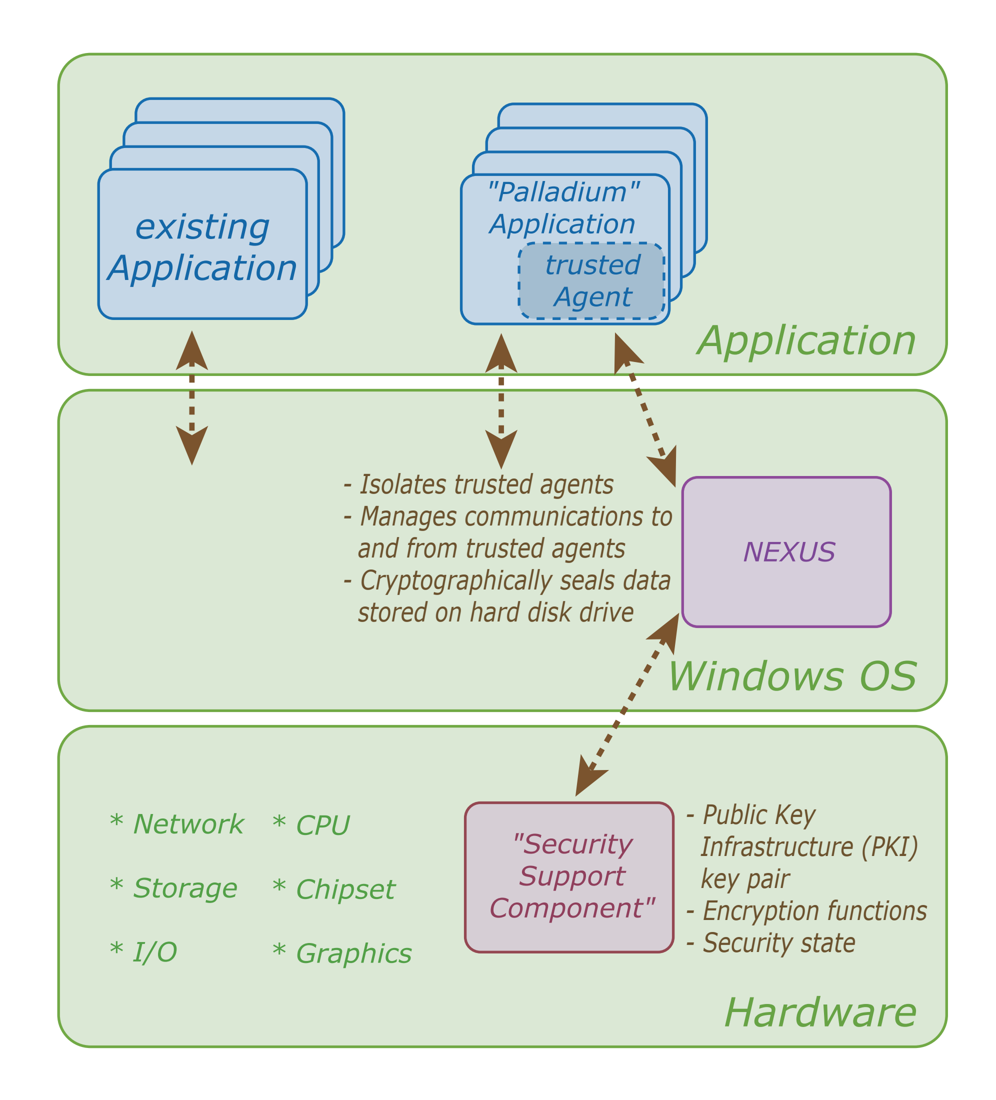


Abbildung 2.1: Funktionsweise von NGSCB

Die „linke“ Seite wird die bisherigen Aufgaben übernehmen, das heißt alle bestehenden Programme werden hier eine gewohnte Umgebung vorfinden, um auch weiterhin ohne Probleme ausgeführt werden zu können. Microsoft ist gezwungen, hier einen Kompromiss einzugehen und die Abwärtskompatibilität bereits existierender Software zu gewährleisten, mit der Einschränkung, dass ihnen die Features, die Nexus bietet, nicht zur Verfügung stehen werden. Um die Vorteile von NGSCB nutzen zu können, müssen existierende Programme angepasst werden oder von Grund auf neu geschrieben werden. Diese – ob nun aus dem Hause

Microsoft stammend oder nicht – werden als „Trusted Agents“ (TA) bezeichnet.

Die „rechte“ Seite wird den neuen Kernel mit dem wohl klingenden Namen „Nexus“ enthalten. Nexus soll eine vor Manipulationen sichere Umgebung für Applikationen schaffen und den Anwender vor Viren, Trojanern und Hacker-Attacken schützen. Es stellt ein Opt-In System dar, das heißt, Nexus kann bei Bedarf geladen und auch wieder entfernt werden. Allerdings funktioniert Nexus nur zusammen mit einem auf der Southbridge oder direkt in der CPU eingebauten Kryptochip namens TPM (Trusted Plattform Module). Nur vertrauenswürdige Programme dürfen hier gestartet werden. Diese sollen – so der Vorschlag von Microsoft – von einer unabhängigen Instanz zertifiziert werden. Die Zertifikate werden auf besagtem Kryptochip gespeichert. Sollte eine Applikation gestartet werden, die das TPM nicht verifizieren kann, weil sie neu ist, besteht die Möglichkeit, dass eine Internetverbindung hergestellt wird und der benötigte Schlüssel von einem Server heruntergeladen werden kann. Warum aber genau diese Verbindung sicher sein soll, führt Microsoft nicht aus.

Der „Nexus“ ist der wichtigste Bestandteil in der vorgestellten Next-Generation Secure Computer Base. Er stellt einen sicheren Bereich im Arbeitsspeicher und auf der Festplatte zur Verfügung – eine Art verbesserte und Hardware-abhängige Sandbox, in der die Anwendungen laufen. Das System soll in der Lage sein, jede gestartete Anwendung auf ihre Integrität zu überprüfen. Stellt Nexus Manipulationen an der Software fest, blockiert es diese. Sofern der Hersteller einer Software dies wünscht, kann er via Internet eine Authentifizierung seines Programms vornehmen lassen. „Nexus“ ist übrigens in der ersten Generation noch nicht multikaskingfähig, für jede geschützte Anwendung muss ein eigener „Nexus“ Kernel gestartet werden.

Microsoft arbeitet bei seiner Palladium-Initiative eng mit Hardware-Herstellern zusammen, die sich in der Trusted Computing Group (TCG) zusammengeschlossen haben. Um die Namens-Verwirrung perfekt zu machen, ist die TCG das Nachfolgegremium der *Trusted Computing Platform Alliance (TCPA)*, die ursprünglich die Rahmenbedingungen und technischen Details der neuen Kryptochips festlegen sollte. Microsoft ging die Entwicklung der TCPA aber in die falsche Richtung, weshalb es die Nachfolge-Organisation initiierte.

In der TCG befinden sich unter anderem Atmel, Infineon, National Semiconductor, Nokia, Philips, Phoenix Technologies, Sony, ST Microelectronics, Verisign und Wave Systems, vor allem aber AMD, IBM, Intel und natürlich Microsoft.

Noch ein Wort zu dem Kryptochip. Im TPM steckt ein 8-Bit-RISC-Prozessor mit einer Taktfrequenz von 33MHz. Er ist in der Lage, einen 2048 Bit langen RSA-Schlüssel in einer halben Sekunde zu berechnen. Zwischen zehn und zwanzig Schlüssel speichert der Chip intern. Der erste Schlüssel wird vom Hersteller speziell für das Mainboard generiert und mit dessen Schlüssel signiert, dieser wiederum mit einem Master-Key der TCPA signiert wurde. Weitere Schlüssel werden dazu benutzt, um den Benutzer beim ersten Rechnerstart zu identifizieren. Der Anwender muss seine Schlüssel wiederum von einer unabhängigen Prüfstelle signieren lassen. Schon jetzt ist abzusehen, dass dieser Prozess sehr aufwendig

und vermutlich auch preisintensiv sein wird.

2.3.3 Kritikpunkte

Um die Kritikpunkte näher zu beleuchten, möchte ich den australischen Professor Bill Caelli von der Technischen Universität in Queensland und Experte für IT-Sicherheit und Kryptografie zitieren. Seiner Meinung nach stellt das von Microsoft und anderen Unternehmen entwickelte Sicherheitsprojekt „Nexus“ eine weit größere Bedrohung dar, als Viren oder Hacker. Auf einer Konferenz in Sydney griff Caelli Microsoft und Intel scharf an.

Demzufolge sei der derzeitige Versuch der Inhalte- und Urheberrechtsinhaber-Lobby, die Kontrolle über die Hardware der Anwender zu erlangen, die größte unmittelbare Bedrohung für die Zukunft der Informationstechnologie.

Caelli ist der Auffassung, dass ein Ziel von NGSCB das urheberrechtliche „dicht“ machen künftiger Intel-basierte Hardware-Systeme ist. „Technisch vermuten wir, dass Intel mit Microsoft zusammenarbeitet, um einen neuen geschützten Operations-Bereich innerhalb des Pentium-Prozessors oder einem seiner untergeordneten Chips einzuführen. Dieser trägt den Codenamen Ring-0“, so Caelli.

„Ich denke, NGSCB wird ein ’sub-operating System’ enthalten, das für den Anwender nicht mehr zugänglich sein wird. Dieses wird über ein neues Hardware-Add-on in Zusammenhang mit Intel-Prozessoren laufen. Während man Nexus für sichere Anwendungen nutzen kann, wird es auch Digital-Rights-Management-Systeme enthalten“, sagt Caelli.

Diese Techniken stellen Caelli zufolge das gesamte Konzept des Besitzverhältnisses des eigenen PCs in Frage. „Stellen Sie sich vor, Sie haben ein Auto erworben und die Motorhaube wurde verschweißt, damit Sie keinen Blick auf den Motor werfen dürfen – es würde zu massiven Protesten kommen... Aber in der IT-Industrie kommen sie [die Softwarehersteller] damit davon und erneut gehen Freiheiten verloren. Was wirklich wichtig an der Angelegenheit ist, ist die Tatsache, dass Inhalteanbieter zum ersten Mal in der Lage sein werden, das Gerät zu kontrollieren – in der Vergangenheit gab es keine derartige Kontrolle; es gab keine Hersteller von LPs, die versucht haben, die Kontrolle über die Plattenspieler zu erlangen. Die gesamte Philosophie wird sich wandeln. In drakonischer Art und Weise, denn es werden die Eigentumsrechte geändert“, so Caelli.

Daneben greift der Professor auch den massiven und erfolgreichen Lobbyismus der Industrie an. „Die Lobby-Arbeit der Inhalteanbieter ist unglaublich. Wie kommt es, dass die Industrie Gehör bei der Regierung findet und die IT-Sicherheit nicht? Im Moment sagt die Inhalte-Industrie, dass sie selbst nur wenig oder überhaupt nichts unternehmen will, um ihr Eigentum zu schützen. Sie wollen Schutz, aber sie wollen nicht dafür zahlen. Warum sollten sie es uns auferlegen, auf sie aufzupassen? Es ist Zeit, dass die Rechtsprechung der Industrie sagt, auf sich selber acht zu geben. So läuft es bei gedruckten Werken seit Hunderten von Jahren - der Urheberechtsinhaber muss sinnvolle Maßnahmen ergreifen, um seine Interessen

zu schützen“, sagt Caelli.

Als Schutzmaßnahme für digitale Inhalte schlägt der Professor vor, bereits bestehende Techniken einzusetzen, um die Verbreitung von Inhalten wie Musik oder ähnlichem zu kontrollieren. So sei zum Beispiel der Einsatz von Wasserzeichen, die dem Käufer eindeutig zugeordnet sind wesentlich sinnvoller, als es Unternehmen wie der RIAA, Disney, MIPI oder Microsoft zu erlauben, den Rechner zu beeinflussen. „Aus technischer Sicht wäre es durchaus machbar für einen Musikanbieter, mir den Download eines aktuellen Albums so zu erlauben, dass die Daten des Kaufs mit den von mir gesetzten Parametern personalisiert werden. Wenn ich diese dann auf meinen Internetserver kopiere, kann ich über dieses Wasserzeichen zurückverfolgt werden.“

2.4 Microsofts Produktaktivierung

„Softwarepiraterie ist ein weltweites Problem, das Software-Entwickler, Händler, Supportexperten und vor allem Käufer schädigt.“ Mit diesem Satz rechtfertigt Microsoft ihre neu eingeführte Praxis der Produktaktivierung in Windows XP, Office XP und Visio 2002.

Laut der Business Software Alliance (BSA) – eine von Microsoft ins Leben gerufene Organisation – betrugen die weltweiten Verluste durch Piraterie im Jahr 2000 schätzungsweise 12 Milliarden Dollar. In Deutschland betrug der Schaden 1,27 Milliarden DM. Zweifelhaft ist jedoch, ob alle Benutzer von Raubkopien diese auch wirklich kaufen würden. Aus diesem Grund stammt die Zahl wohl eher aus einem „Was-wäre-Wenn-Szenario“. Microsoft geht hier sogar noch einen Schritt weiter und sieht sich offensichtlich in der Rolle der Executive. So kann man auf der Homepage des Unternehmens folgendes lesen:

Bereits seit Juni 1993 sind in Deutschland Computerprogramme ausdrücklich urheberrechtlich geschützt. Damals wurde basierend auf einer EU-Richtlinie vom Mai 1991 der Schutz von Computerprogrammen auch im deutschen Urheberrecht verbessert.

So bestimmt § 69 a Absatz 3 UrhG:

Computerprogramme werden geschützt, wenn sie individuelle Werke in dem Sinne darstellen, dass sie das Ergebnis der eigenen geistigen Schöpfung ihres Urhebers sind. Zur Bestimmung der Schutzfähigkeit sind keine anderen Kriterien, als qualitative oder ästhetische Gesichtspunkte anzuwenden.

Hier wird es deutlich: Unrechtmäßige Vervielfältigung von Software und deren Einsatz ist kein Kavaliersdelikt!

Abbildung 2.2: Auszug aus der Microsoft-Webseite über die Produktaktivierung

Nach dem Erwerb eines Microsoft Produktes muss man dieses seit neuestem also aktivieren. Was bedeutet das genau? Die Installation der Software läuft wie gewohnt ab. Nach dem Einlegen der CD und dem Start des Setups wird man nach der Eingabe des Produkt-Keys gefragt, der in der Regel auf der CD selber oder einem beiliegenden Handbuch abgedruckt ist. Er besteht aus 5 – durch Bindestriche getrennte – fünfstellige Zahlen- und Buchstabenblöcke. Während der Installation wird aus diesem Produkt-Key eine eindeutige Produkt-ID gebildet. Wie genau das geschieht, ist nicht bekannt. Eine typische Produkt-ID hat folgende Form: 12345-123-1234567-12345.

Der Benutzer kann das soeben erfolgreich installierte Microsoft-Produkt nun einsetzen – mit einer Einschränkung: Windows XP läuft 30 Tage, Office XP und Vision 2002 stellen ihre Arbeit nach 50-maligem Start ein. Danach wird eine Aktivierung erforderlich. Der Kunde

ist gezwungen, ja sogar genötigt, bei einer Hotline anzurufen, um einen Freischaltcode zu erfahren oder kann die Aktivierung über das Internet abwickeln.

In jedem Fall – ob nun telefonisch oder elektronisch – wird eine 50-stellige Zahl, die so genannte Installations-ID, an Microsoft übermittelt. Die Installations-ID wird aus der oben genannten Produkt-ID und einem „Hardwarehash“ gebildet. Das heißt also, dass die Aktivierung durch den „Hardwarehash“ an einen bestimmten Rechner gebunden ist. Dieser besagte Hash ist es auch, der die Produktaktivierung Microsofts zu einem zu untersuchenden Thema für die vorliegenden Arbeit macht. Leider sind die Informationen, die Microsoft über die genauen Vorgänge herausgibt, nicht sehr umfangreich. Fest steht jedoch, dass der die Komponenten identifizierende Hash genau acht Byte lang ist – oder 64 Bit. Über jede, der in *Tabelle 2.1* aufgeführten Hardwarekomponenten, wird ein 128 Bit langer Hashwert gebildet. Von diesem Hash fließt ein bestimmter Teil in den endgültigen „Hardwarehash“ ein. Welcher Anteil und wie genau, ist leider nicht bekannt.

Nr.	**Name der Komponente**	**Beispielhashwert** (Anzahl der Bits)
1	Grafikkarte	00010 (5)
2	SCSI-Adapter	00011 (5)
3	IDE-Adapter	0011 (4)
4	MAC-Adresse des Netzwerkadapters	1001011000 (10)
5	RAM-Bereich (z. B. 0-64 MB, 64-128 MB usw.)	101 (3)
6	Prozessortyp	011 (3)
7	Seriennummer des Prozessors	000000 (6)
8	Festplattenlaufwerk	1101100 (7)
9	Seriennummer des Festplattenlaufwerkes	1001000001 (10)
10	CD–ROM / CD-RW / DVD-ROM	0101111 (7)
-	„Andockbar“	0 (1)
-	Hardwarehash-Version (Version des verwendeten Algorithmus)	001 (3)

Tabelle 2.1: *Werte für Hardwarehash-Komponenten*

Zusammengesetzt ergeben die 12 „Teilhashes“ den zur Bildung der Produkt-ID benötigten „Hardwarehash“. Die in dieser Aufzählung an elfter Stelle stehende „Komponente“ meint in Wirklichkeit eine Eigenschaft. Sie bezieht sich dabei auf die Fähigkeit von Laptops in einer Dockingstation benutzt werden zu können oder PCMCIA-Karten zu akzeptieren

(eine Dockingstation oder PCMCIA-Karten sind für den Fall gedacht, dass Hardware nicht mehr angezeigt wird oder fehlende Geräte darauf hindeuten, dass Hardwareänderungen vorgenommen wurden).

Schließlich enthält der „Hardwarehash“ eine Versionsnummer, die den verwendeten Algorithmus identifizieren soll. Zusammen mit den anderen verwendeten Werten, die allgemeiner Art sind, könnten zwei verschiedene Computer – wenn auch unwahrscheinlich – denselben Hardwarehash erstellen.

Die Product ID (neun Byte) und der Hardwarehash (acht Byte) werden von Microsoft zum Verarbeiten der Aktivierungsanforderung verwendet. Wird die Aktivierung über das Internet durchgeführt, bilden diese beiden Werte die Installations-ID (in einem Binärformat) und werden zusammen mit den Header-Informationen der Anforderung in einem Binärformat über Secure Sockets Layer (SSL in HTTP) direkt an das Aktivierungssystem von Microsoft gesendet. Bei der Internetaktivierung finden drei Kommunikationsvorgänge statt:

1. **Handshakeanforderung** enthält die Product-ID, den „Hardwarehash“ und Zusatzinformationen, wie zum Beispiel die Session-ID und die Version der Aktivierungstechnologie, insgesamt 262 Byte.

2. **Lizenzanforderung** enthält die Product-ID, den „Hardwarehash“ und die optionalen Kundendaten, um gegebenenfalls freiwillige Registrierungsinformationen zu speichern. Wird die Registrierung übersprungen, ist diese Struktur leer. Sie enthält zudem die Session-ID und das PKCS-#10-Zertifikat zum Verschlüsseln. Die PKCS-#10-Struktur kann leicht variieren (insgesamt etwa 2763 bis 3000 Byte), je nachdem, ob freiwillige Registrierungsinformationen angegeben wurden, oder nicht. Der Aufbau eines solchen Zertifikates is in [RSA00] beschrieben.

3. **Bestätigungsanforderung** enthält die Zertifikats-ID (wird nach der Lizenzanforderung an den Computer des Benutzers zurückgegeben), das Ausgabedatum und den Fehlercode, insgesamt 126 Byte.

Ist die Internetaktivierung erfolgreich, wird die Aktivierungsbestätigung als digitales Zertifikat direkt zurück an den Computer des Benutzers gesendet. Dieses Zertifikat wird von Microsoft digital signiert, so dass es nicht geändert oder gefälscht werden kann. Das Bestätigungspaket, das bei der Internetaktivierung zurückgegeben wird, ist etwa 9 KB groß (die digitale Zertifikatskette macht den Großteil der Paketgröße der Bestätigungsdaten aus). Das erklärt auch, warum per Internetaktivierung deutlich mehr Daten über die Leitung gehen, als bei der Variante per Telefon.

Wählt der Kunde die Aktivierung per Telefon wird ihm die Installations-ID auf dem Bildschirm angezeigt. Allerdings wird sie, anders als bei der Internetaktivierung, als 50-stellige Dezimalzahl dargestellt. Umfragen und Testläufe haben ergeben, dass bei der verbalen Kommunikation weniger Fehler entstehen, als wenn man die kürzere Variante mit alphanumeri-

schen Zeichen wählen würde. Zusätzlich werden die Daten verschlüsselt, um unbeabsichtigte Mithörer auszuschalten. Eine Prüfsumme stellt die Integrität der Installations-ID sicher.Die telefonische Aktivierung setzt sich aus vier Schritten zusammen:

1. Der Kunde wählt das Land aus, in dem er sich befindet, damit ortsgültige Telefonnummern angezeigt werden können.
2. Wählen der Telefonnummer.
3. Weitergeben der Installations-ID an den Kundendienstmitarbeiter.
4. Eingeben der Bestätigungs-ID, die vom Kundendienstmitarbeiter angegeben wird.

Die Bestätigungs-ID ist eine 42-stellige ganze Zahl, die den Aktivierungsschlüssel und die Prüfziffern enthält, die bei einer Fehlerbehandlung hilfreich sind. Gibt der Kunde diesen Code bei sich ein und bestätigt ihn, ist das Produkt freigeschaltet.

Als Microsoft erstmals ihre Pläne für die Aktivierung von Software vorstellte, gingen viele Datenschutzverbände auf die Barrikaden. In Deutschland und den anderen EU-Staaten ist das Datenschutzgesetz sehr viel strenger als in den USA – dem Sitz von Microsoft. Daher trat die Firma dem Safe-Harbor-Abkommen bei und verpflichtete sich damit, *„personenbezogene Daten europäischer Kunden in einer dem Datenschutzniveau der EU angemessenen Weise zu behandeln“*. Bei den Safe Harbor Principles handelt es sich um ein Abkommen, das zwischen der Europäischen Kommission und dem amerikanischen Department of Commerce geschlossen wurde, um den Schutz europäischer personenbezogener Daten, die an amerikanische Unternehmen übermittelt werden, zu gewährleisten.

In einem Urteil des Oberlandesgerichtes München vom 12. Oktober 2000 wurde ein unerwarteter Registrierungszwang nach der Installation für unzulässig erklärt. Microsoft bestätigt aber, dass es sich bei der Produktaktivierung – und das ist entscheidend – eben nicht um eine Registrierung persönlicher Daten handelt. Sie ist, und das ist gesetzlich gefordert, absolut anonym. Das hat auch die TÜV Informationstechnik GmbH Essen bestätigt.

Ein weiterer Gesichtspunkt, den es zu beachten gilt, ist die BGH-Entscheidung vom Juli 2000. Es ist richtig, dass die Produktaktivierung an die Hardware gebunden ist, um die unerlaubte Mehrfachnutzung zu verhindern. Im Gegensatz zu der Entscheidung ist die Aktivierung aber nicht an einen bestimmten Computer gebunden. Dem Kunden steht es jederzeit frei, das von ihm erworbene Produkt mit samt der Lizens wieder zu verkaufen. Der Erwerber kann (und muss) die Software dann erneut aktivieren. Die Verkehrsfähigkeit des Software-Produkts wird damit durch die Aktivierung nicht beeinträchtigt.

Kapitel 3

Verschlüsselung und Signaturen

Um die Kommunikation und die Übertragung von Daten in der Game-Feature-Plattform abzusichern, werden verschiedene Verschlüsselungsalgorithmen benutzt. Aus diesem Grund soll in diesem Kapitel ein kurzer Einblick in die Kryptologie erfolgen. Dabei wird auf die zu Grunde liegende Mathematik verzichtet und lediglich ein grober Überblick gewährt. Für tiefer gehende Recherche sei ausdrücklich auf einschlägige Literatur verwiesen. Als gute Ergänzung zu diesem Kapitel bietet sich das Buch „Abenteuer Kryptologie. Methoden, Risiken und Nutzen der Datenverschlüsselung“ ([WOB98]) an.

Sensible Softwaresysteme können vielen Arten von sicherheitskritischen Bedrohungen unterworfen sein. Daraus ergeben sich eine Reihe genereller Anforderungen an Sicherheitssysteme:

- **Vertraulichkeit (Confidentiality)** Hiermit soll ausgedrückt werden, dass Daten und Dienste eines Systems nur berechtigten Nutzern zugänglich gemacht werden sollen. Dies wird in der Regel durch Autorisation / Identifikation des Benutzers bzw. kryptografische Verfahren wie z.B. symmetrische oder asymmetrische Verschlüsselung erreicht.

- **Integrität (Integrity)** Diese Anforderung bezieht sich auf die Unversehrtheit der genutzten Daten bzw. die korrekte Arbeitsweise der angeforderten Dienstleistung. Gängige Verfahren zur Gewährleistung dieser Anforderung sind z.B. digitale Signaturen und Prüfsummen.

- **Verfügbarkeit (Availability)** ist die Eigenschaft, eine Dienstleistung dem Benutzer immer dann zur Verfügung stellen zu können, wenn dieser sie wünscht.

Des Weiteren existieren eine Reihe weiterer Anforderungen, welche insbesondere in Zusammenhang mit der Nutzung des Internets ihre Bedeutung finden. Unter **Zurechenbarkeit** versteht man die Eigenschaft, jede Aktion und deren Ergebnisse einem konkreten Nutzer zuordnen zu können. Dies ist konträr zur Forderung nach **Anonymität**. Eine weitere Rolle

spielt die **Verbindlichkeit**, welche es ermöglichen soll, nach einer durchgeführten Aktion den Urheber und somit Verantwortlichen einwandfrei zu identifizieren.

Für alle diese Anforderungen existieren eine Reihe von sicheren Lösungen, welche im Folgenden näher erläutert werden sollen. Dabei bedeutet „sicher“ immer einen Zustand relativ zum momentanen Stand der verfügbaren Rechentechnik und relativ zum Stand der Erkenntnisse in der Kryptoanalyse (Wissenschaft, welche sich mit der Entschlüsselung befasst). Ein Maß für die Sicherheit ist der Aufwand (Rechenleistung, Zeit, ...) der benötigt wird, um einen jeweiligen Sicherheitsmechanismus zu umgehen.

3.1 Symmetrische Verschlüsselung

Ein Verschlüsselungsverfahren wird als symmetrisch bezeichnet, wenn der gleiche Schlüssel sowohl zum Ver- als auch zum Entschlüsseln verwendet wird. Der Schlüssel, da für beide Richtungen benutzt, muss demnach sowohl Sender als auch Empfänger bekannt sein und zu diesem Zweck vorher persönlich ausgetauscht werden. *Abbildung 3.1* zeigt die generelle Vorgehensweise.

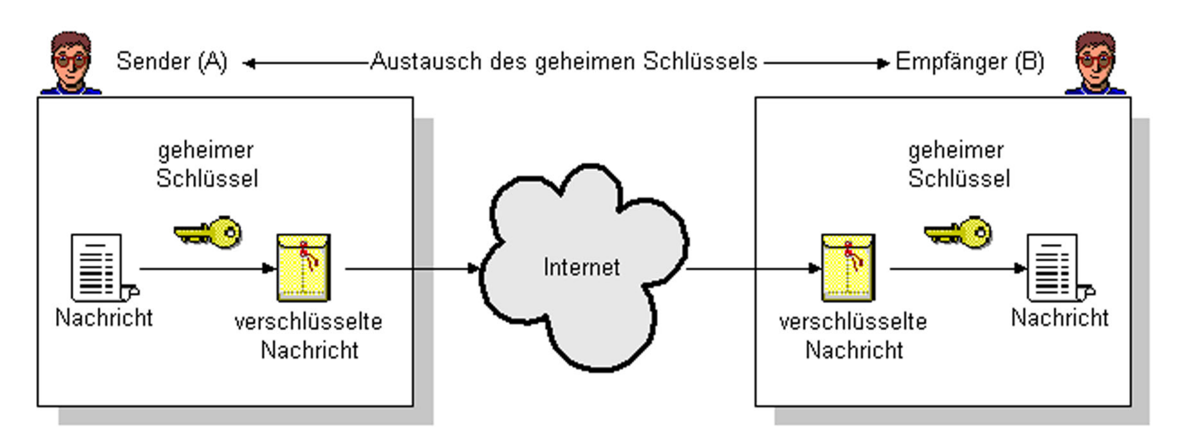


Abbildung 3.1: Symmetrische Ver- und Entschlüsselung

Man teilt die symmetrischen Verfahren in Blockchiffren und Stromchiffren auf. Mit Stromchiffren wird der Quelltext Zeichen für Zeichen ver- bzw. entschlüsselt, um den Zieltext zu erhalten. Ein Blockchiffre arbeitet mit einer festen Blockgröße und ver- bzw. entschlüsselt mehrere Zeichen in einem Schritt.

Symmetrische Verschlüsselungsverfahren sind schon sehr lange bekannt und fanden bereits Erwähnung zu Julius Cäsars Zeiten. Effizient implementiert können sie extrem schnell sein und mehrere Megabyte Daten pro Sekunde ver- bzw. entschlüsseln. Trotz des vergleichsweise kurzen Schlüssels erreichen sie eine hohe Sicherheit. Allerdings haben Verfahren, die auf der symmetrischen Verschlüsselung basieren auch einen entscheidenen Nachteil. Wie bereits angedeutet, muss der Schlüssel beiden Parteien bekannt sein. Idealerweise muss er auch regelmäßig gewechselt werden. Ist der Schlüssel einem Angreifer bekannt, ist es für ihn ein Leichtes an Information zu gelangen und Fehlinformationen durch Veränderung der

Originalnachricht zu verbreiten. Ein weiteres typisches Problem ergibt sich bei der Frage, wie der Schlüssel ausgetauscht werden kann. Dafür wäre ein sicherer Kanal nötig. Dafür wiederum wäre aber ein geheimer Schlüssel notwendig. . .

3.2 Asymmetrische Verschlüsselung

Ein weiteres Verschlüsselungsverfahren ist die asymmetrische Verschlüsselung. Sie basiert auf der Verwendung eines zusammengehörenden Schlüsselpaares, wobei ein Schlüssel zur Ver- und ein anderer zur Entschlüsselung genutzt wird. Es ist aber nicht (bzw. nicht mit vertretbarem Aufwand) möglich, aus der Kenntnis des einen Schlüssels den anderen zu berechnen. Aus diesem Grund kann der Schlüssel zur Verschlüsselung veröffentlicht werden. Die beiden Schlüssel bezeichnet man als öffentlichen (public) und geheimen (private) Schlüssel. Daher leitet sich auch der Name *Public-Key-Algorithmus* ab. Das bedeutet, dass ein Schlüssel veröffentlicht werden kann, ohne die Sicherheit der zu übertragenden Nachricht zu gefährden.

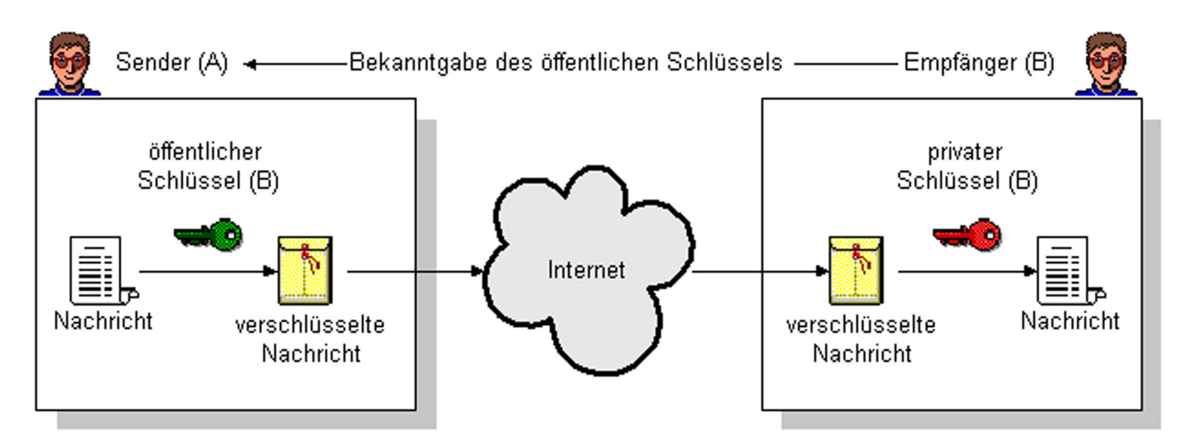


Abbildung 3.2: Asymmetrische Ver- und Entschlüsselung

Asymmetrische Verfahren sind ein relativ neues Gebiet der Kryptografie und entstanden erst in den 70er Jahren des letzten Jahrhunderts. Der wohl bekannteste Vertreter ist der 1977 entwickelte RSA-Algorithmus. Er wurde nach seinen Erfindern Ronald L. Rivest, Adi Shamir and Leonard M. Adleman benannt.

Die asymmetrische Verschlüsselung kann auch genutzt werden, um das Problem der Authentifizierung zu lösen (siehe *Abbildung 3.3*). Zu diesem Zweck werden die öffentlichen Schlüssel von Sender und Empfänger gegenseitig bekannt gemacht. Der Sender verschlüsselt die Nachricht zunächst mit seinem eigenen privaten und dann mit dem öffentlichen Schlüssel des Empfängers. Nach Erhalt der Nachricht entschlüsselt der Empfänger diese zunächst mit seinem privaten und dann mit öffentlichen Schlüssel des Senders. Der letzte Schritt ist jedoch nur dann erfolgreich, wenn die Nachricht wirklich von dem bezeichneten Sender kam, da andernfalls der verwendete öffentliche Schlüssel nicht passend ist.

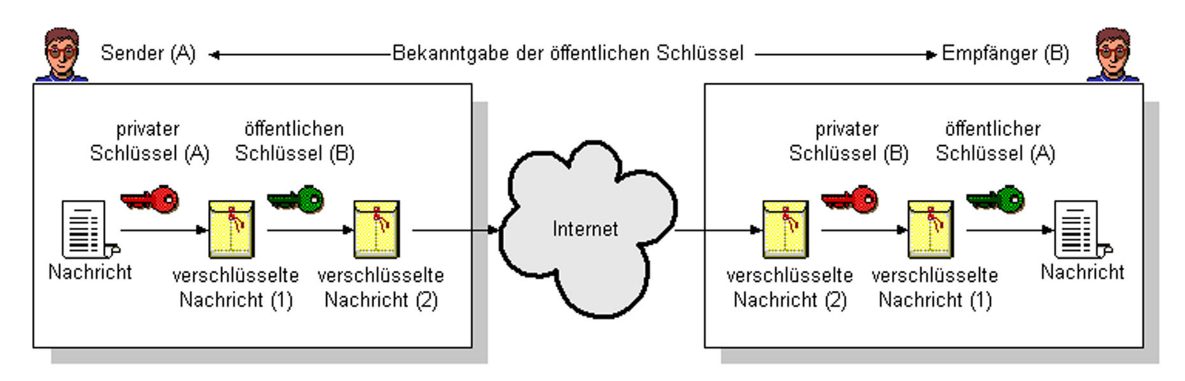


Abbildung 3.3: Asymmetrische Ver- und Entschlüsselung mit Authentifizierung

Die Vorteile liegen auf der Hand. Nur der geheime Schlüssel muss auch wirklich geheim gehalten werden. Der öffentliche Schlüssel kann jedem zur Verfügung gestellt werden. Allerdings muss dessen Authentizität sichergestellt werden. Abhängig vom eingesetzten Verfahren kann ein Schlüsselpaar – im Gegensatz zur symmetrischen Methode – eine sehr hohe Lebensdauer haben, oftmals sogar Jahre. Nachteilig kann sich die deutlich längere Rechenzeit zur Ver- und Entschlüsselung auswirken. Die Schlüssel sind relativ lang und keine der heute bekannten Verfahren zur asymmetrischen Kodierung kann als wirklich sicher bezeichnet werden, da sie auf so genannten mathematischen Einwegfunktionen beruhen. Sie definieren sich dadurch, dass sie nur in eine Richtung einfach zu berechnen sind. Der Beweis dafür steht allerdings noch aus.

3.3 Hybride Verfahren

Beide Verfahren haben ihre Vor- und Nachteile. Aus diesem Grund wird oftmals eine Kombination aus symmetrischer und asymmetrischer Verschlüsselung genutzt.

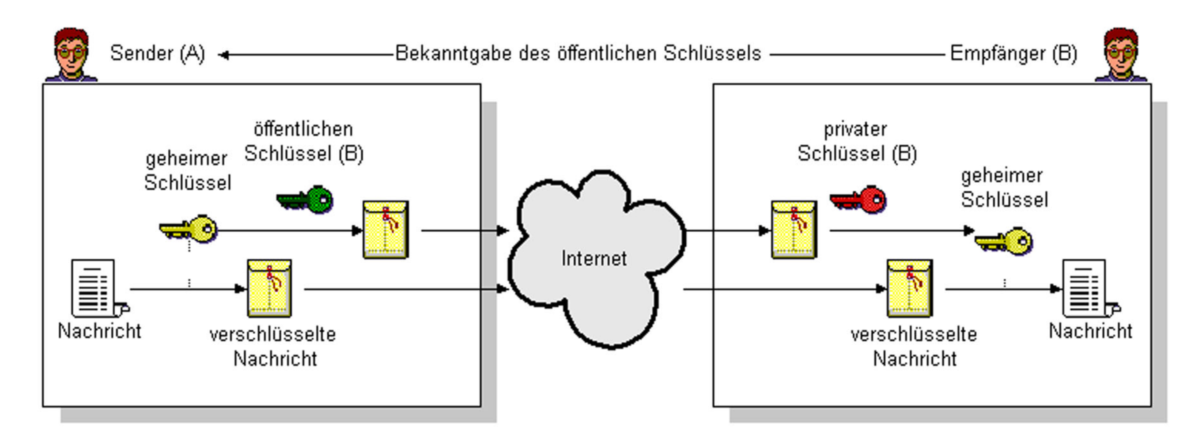


Abbildung 3.4: Hybrides Verfahren zur Ver- und Entschlüsselung

Dabei wird eine Nachricht durch den Empfänger zunächst mit einem speziellen geheimen Schlüssel (Session Key) symmetrisch verschlüsselt. Anschließend wird dieser Schlüssel mit dem öffentlichen Schlüssel des Empfängers asymmetrisch verschlüsselt und übertragen.

Der Empfänger kann nun asymmetrisch mit seinem privaten Schlüssel den Session Key und somit die eigentliche Nachricht symmetrisch entschlüsseln. Da nur der symmetrische Schlüssel verschlüsselt wird, bleibt der Rechenaufwand bei der asymmetrischen Verschlüsselung relativ gering. *Abbildung 3.4* verdeutlicht noch einmal die Vorgehensweise.

3.4 Digitale Signatur

Um die Integrität und Authentizität der übermittelten Daten zu gewährleisten, d.h. um sicherzustellen, dass Sender und Empfänger über die gleichen Daten verfügen und diese nicht etwa während des Transfers modifiziert wurden, kann das Verfahren der digitalen Signatur angewendet werden. Dabei wird mittels einer speziellen Einweg-Hash-Funktion der so genannte digitale Fingerabdruck (Message Digest) einer Nachricht ermittelt, welcher dabei folgende Eigenschaften aufweist:

- Er ist relativ leicht aus dem Dokument zu berechnen
- Er lässt keinen Schluss auf die ursprüngliche Nachricht zu
- Eine zweite Nachricht mit demselben digitalen Fingerabdruck lässt sich nur mit sehr hohem Aufwand erzeugen

Der Message Digest wird anschließend mit dem privaten Schlüssel des Senders verschlüsselt und zusammen mit der (ebenfalls verschlüsselten) Nachricht übertragen. Der Empfänger entschlüsselt nun den übertragenen Hash-Wert mit dem öffentlichen Schlüssel des Senders, wodurch die Authentizität des Senders gewährleistet wird. Anschließend berechnet er aus der erhaltenen Nachricht einen eigenen Message Digest und vergleicht diesen mit dem von dem Empfänger erhaltenen. Stimmen beide überein, verfügen Sender und Empfänger über dieselbe Nachricht (siehe *Abbildung 3.5*).

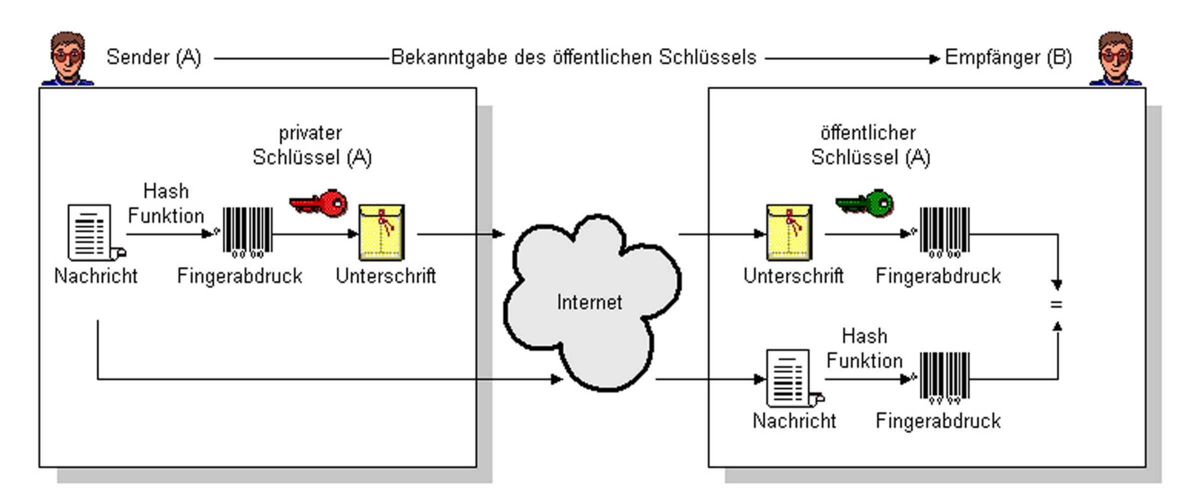


Abbildung 3.5: Digitale Signatur

Zu den bekanntesten Hash-Algorithmen gehören der 1991 von Ron Rivest entwickelte *Message Digest 5 (MD5)* Algorithmus und der *Secure Hash Algorithm 1 (SHA-1)*.

Kapitel 4

Game-Feature-Plattform

Ausgangspunkt dieser Arbeit ist die bereits existierende Game-Feature-Plattform der 4FriendsOnly.com Internet Technologies AG[1]. Die Firma wurde 2000 von Dr. Ing. Jürgen Nützel gegründet. Im Mittelpunkt der Geschäftsaktivitäten stehen virtuelle Waren. Das Unternehmen vertreibt die verschiedensten Produkte bzw. Dienstleistungen:

- Die **Game-Feature-Plattform** für unabhängige Spieleentwickler.
- Das Micro-Paymentsystem **Paybest** als Bezahlsystem für digitale Waren.
- Das **PotatoSystem** als Vertriebsnetzwerk für Dateien im Internet.

Obwohl die Businessmodelle für Shareware erfolgreich sind, ist man bei 4FriendsOnly.com bestrebt, neue Möglichkeiten für den Vertrieb von Software zu erschließen. Zum einen bietet sich das „Pay-per-Use“ Modell an. Üblicherweise kommt diese Form mit einem Abonnement daher. Der Kunde zahlt eine feste Teilnahmegebühr, um das Produkt nutzen zu dürfen. Bekannte Vertreter sind so genannte *Massive Multiplayer Online RPG (RPG steht für Roll-Play Game)*, zum Beispiel Eve Online: The Second Genesis[2] oder Star Wars Galaxy: An Empire Divided[3]. Bei beiden wird eine monatliche Gebühr fällig. Als nachteilig stellt sich hier die hohe Eintrittsschwelle heraus. Ein Spielen ohne Abonnement ist nicht möglich. Um trotzdem neue Kunden gewinnen zu können, werden von den Entwicklern Trial-Accounts vergeben. Diese gestatten es dem Kunden, das Spiel für eine bestimmte Zeit (oftmals 7 oder 14 Tage) völlig kostenlos auszuprobieren.

Ein anderes System – das von 4FriendsOnly.com favorisierte – ist das „Pay-per-Feature“-Businessmodell. Hierbei bestimmt der Spieler den Preis. Das Basissystem ist frei erhältlich. Das wiederum garantiert einen hohen Verbreitungsgrad der Software. Der Kunde entscheidet selber, wie viel Geld er in das Spiel investieren möchte. Ihm steht es frei, das Spiel durch

[1] `http://www.4fo.de/`
[2] `http://www.eve-online.com/`
[3] `http://starwarsgalaxies.station.sony.com/`

zusätzliche, kostenpflichtige Features zu erweitern. Auf der anderen Seite haben Entwickler die Möglichkeit, Updates und Extensions zu einem späteren Zeitpunkt nachzuliefern.

Die patentierte Game-Feature-Plattform (GFP) ist die konsequente Weiterentwicklung des Shareware-Gedankens und die Umsetzung des „Pay-per-Feature“-Businessmodells (Vgl. [NUE01]). Sie soll es in erster Linie Spieleentwicklern erleichtern, ihr Produkte schnell und effektiv zu vermarkten. Die Plattform steht aber auch allen anderen offen, die Software vertreiben wollen. Zielgruppe sind die Low-Budget Produkte (gewöhnlicherweise Shareware-Programme), die nicht viel in den Vertrieb ihrer Produkte investieren möchten, aber trotzdem einen Obolos für das erstellte Stück Software erhalten möchten. Die Game-Feature-Plattform bietet eine Reihe von zusätzlichen Funktionen:

- **Online-Highscore-Verwaltung**. Eine API-Funktion ermöglicht das Senden von Spielständen. 4FO erstellt die Highscore-Listen, die der Entwickler in unterschiedliche Webseiten integrieren kann.
- Die Entwickler können in Echtzeit ihre **Umsätze kontrollieren**.
- **User-Level-Verkauf**. Bietet der Entwickler einen Level-Editor an, so können die Nutzer ihre eigenen Levels über die GFP anbieten.
- **Kopierschutz**. Nutzer können gekaufte Levels nicht auf andere Rechner kopieren.
- **Service für den User**. Die GFP ermöglicht nach einem PC-Crash die kostenlose Neuinstallation bereits bezahlter Levels.
- Entwickler können über die GFP kostenlose **Patches und Updates verteilen**.

4.1 Funktionsweise

Der Grundgedanke besteht darin, dass das Spiel in einer abgespeckten Version kostenlos im Internet oder zum Selbstkostenpreis auf CD vertrieben wird. Die Basis-Version eines über die GFP publizierten Spieles enthält zum Beispiel nur ein oder zwei Level, ein begrenztes Set an Karten oder Fahrzeugen. So ist es dem Spieler möglich, die Funktionsweise ohne Zeitbegrenzung zu testen. Auf Wunsch können dann kostenpflichtige Erweiterungen hinzugeladen werden. Ähnlich wie bei dem Shareware-Prinzip kann der Kunde also zuerst einmal das Spiel ausprobieren. Gefällt ihm, was er sieht, steht es ihm frei, durch Bezahlung das Spiel zu komplettieren. Hierin liegt auch der Unterschied zu der Shareware. Bei dessen Verwendung wird ein einmaliger Betrag fällig, der zur vollständigen Nutzung des Programms berechtigt. Nicht so bei der Game-Feature-Plattform. Der Kunde kann wählen, welche Erweiterungen er kaufen möchte. In der Regel kosten sie einzeln nur wenige Cents bis hin zu ein paar Euro.

Bei der Game-Feature-Plattform werden dem Benutzer viele kleinere Erweiterungen (Spielelevel, Karten, Einheiten, ...) – so genannte Features – zur Verfügung gestellt. Über

die GFP können sogar Updates oder Bugfixes für das Spiel angeboten werden. Der Erweiterbarkeit sind keine Grenzen gesetzt. Die dabei heruntergeladenen Features werden verschlüsselt auf der Festplatte abgelegt. Die zur Entschlüsselung benötigten Schlüssel sind nur auf diesem einen Rechner gültig. Somit ist die Weitergabe der erworbenen Addons nahezu ausgeschlossen (auf Systemen mit identischer Hardware wäre das erfolgreiche Entschlüsseln der Features durchaus vorstellbar).

Die Game-Feature-Plattform ist ein von 4FriendsOnly.com entwickeltes Client/Server-System, um typische „Pay-per-Feature"-Szenarien zu realisieren. Am 28. November 2002 wurde das zugehörige Patent „Verfahren zur Verfügbarmachung von multimedialen Datenmengen" (Patentnummer: DE 10059230 C2) durch das Deutsche Patent- und Markenamt erteilt.

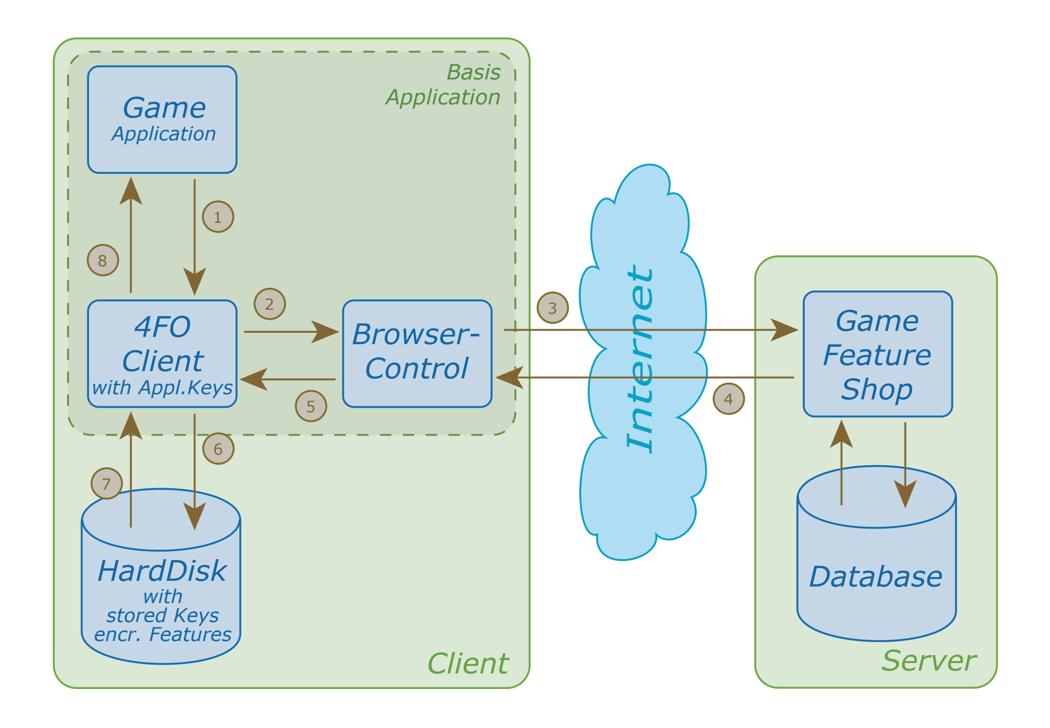


Abbildung 4.1: Architektur der Game-Feature-Plattform

Der grundlegenden Aufbau der Game-Feature-Plattform wird in *Abbildung 4.1* gezeigt. Die Zahlen geben dabei die Reihenfolge des Ablaufs an. Im wesentlichen besteht die GFP aus drei Teilen:

- Basisapplikation mit dem integrierten 4FO-Client
- Micro-Payment-System „Paybest"
- Server mit integriertem Game-Feature-Shop

4.1.1 Der 4FO-Client

Der 4FO-Client ist das Herzstück der Game-Feature-Plattform. Er ist eine Komponente, die als Windows-Bibliothek von 4FriendsOnly.com AG zur Verfügung gestellt wird. Diese DLL muss von den Entwicklern in das Spiel (oder jedes andere beliebige Programm) eingebunden werden. Der 4FO-Client regelt selbstständig die Kommunikation mit dem Game-Feature-Shop (Server). Entscheidet sich der Kunde, die Anwendung um ein oder mehrere Features zu erweitern, wird mittels eines eingebauten Browser-Controls eine sichere Verbindung zum Server aufgebaut. Dazu ist eine Internetverbindung notwendig. Dort bestätigt der Benutzer die zur Auswahl stehenden Features und bezahlt über ein integriertes Bezahlsystem. Der Download und die Installation der verschlüsselten Features erfolgt automatisch durch ein integriertes Protokoll. Die Features werden verschlüsselt auf der Festplatte abgelegt. Die Entschlüsselung für die Windows-Anwendung erfolgt on-the-fly durch den 4FO-Client, sobald die Applikation auf ein Feature zugreift.

4.1.2 Das Micro-Payment-System „Paybest“

Für die Bezahlung der Game-Features stellt die 4FriendsOnly.com AG ein eigenes Micro-Payment-System „Paybest“ bereit. Es besteht aber auch die Möglichkeit, durch Nutzung der Schnittstellen jedes andere Bezahlsystem zu integrieren. Hiermit kann der Nutzer sein Prepaid-Konto in dem Game-Feature-Shop auffüllen. Neben der Möglichkeit per Telefon- oder Handy-Anruf zu bezahlen, hat „Paybest“ auch die Systeme *paysafecard*, *Moneybookers*, *MicroMoney*, *PayPal* und *Firstgate* integriert. Nachdem der Kunde die zu kaufenden Features ausgewählt und bestätigt hat, wird er auf den „Paybest“ Server weitergeleitet. Mit der Weiterleitung werden Sessiondaten und Preis übertragen. Sollte das Konto nicht ausreichend gedeckt sein, besteht die Möglichkeit, über eines der oben genannten Bezahlsysteme eine Transaktion zu veranlassen oder eine bzw. mehrere Gutscheinnummern einzugeben.

Eine Gutscheinnummer ist eine 8-stellige Buchstaben-Ziffern-Kombination und verkörpert einen festgelegten Wert (2,50 Euro). Diese Nummern kann man über eine kostenpflichtige 0190er-Telefonnummer (bzw. 0900) erhalten. Ein Anruf bei besagter Nummer dauert 82 Sekunden und durch eine automatische Ansage erhält man eine Gutscheinnummer. Somit erfolgt die Abrechnung über die Telefonrechnung. Alternativ kann mit einer Prepaid-Karte von Paysafecard oder mit Paypal bezahlt werden. Der 4FO-Payment-Server prüft nun die Gültigkeit und Wertigkeit der eingegebenen Nummern. Sind sie in Ordnung, so wird der übermittelte Preis von den Nummern abgebucht und dem Anbieter (Spieleentwickler) gutgeschrieben. Eine Nummer kann somit mehrmals für kleinere Teilbeträge genutzt werden. Nach erfolgter Transaktion beginnt der Download der verschlüsselten Features. Quittiert der Client den erfolgreichen Empfang, wird der Kunde zu einer Abschlussseite weitergeleitet und das Browser-Control schließt sich (Vgl. [NUE02]).

4.1.3 Der Game-Feature-Shop

Der Game-Feature-Shop ist eine spezialisierte E-Commerce-Plattform und wurde komplett in Java unter Zuhilfenahme von Java-Server-Pages (JSP) umgesetzt.

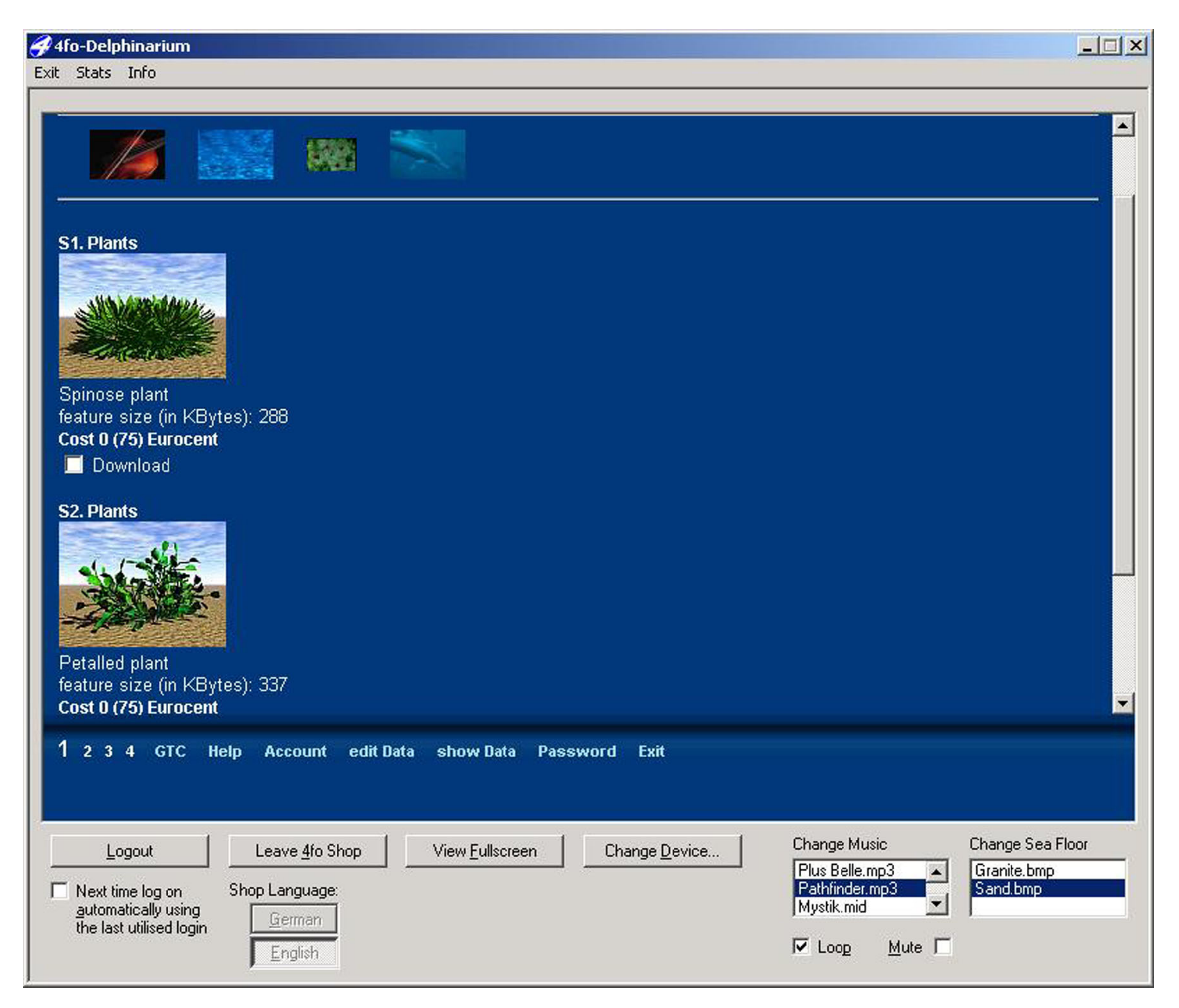


Abbildung 4.2: Game-Feature-Shop

Die *Abbildung 4.2* zeigt ein Beispiel aus dem 4FriendsOnly.com Delphinarium. Dabei ist das hier abgebildete Windowsprogramm die eigentliche Anwendung zur Demonstration der Game-Feature-Plattform. Der Hauptteil (blauer Hintergrund) stellt das Web-Browser-Control mit dem Game-Feature-Shop dar.

Der Shop wird durch das entsprechende Spiel „on-demand“ kontaktiert. Bevor die zur Verfügung stehenden Features angezeigt werden, ist es notwendig, dass sich der Kunde einloggt. Besteht noch kein Login, muss zunächst eines angelegt werden. Die Speicherung der Kundendaten erfolgt in einer SQL-Datenbank. Für jeden Nutzer werden Prepaid-Kontostände verwaltet. Levels oder andere Features können nur bei ausreichend gefülltem Konto heruntergeladen werden. Das Konto kann über ein integriertes Bezahlsystem aufgeladen werden. Die Features werden nach erfolgreicher Transaktion durch den Shop verschlüsselt und auf die Festplatte des Kundens übertragen.

4.2 Anwendungsfälle

In diesem Kapitel sollen mittels zweier Anwendungsfälle das Systemverhalten der Game-Feature-Plattform dargestellt werden. Anwendungsfälle (Use Cases) sind ein Bestandteil der UML. Sie beschreiben, wie verschiedene Akteure auf ein System einwirken und mit dem System interagieren, d.h. sie haben die Aufgabe zu verdeutlichen, was das System leisten soll, aber nicht wie das System etwas leisten soll. *Abbildung 4.3* zeigt was passiert, wenn sich ein neuer Benutzer am System anmeldet.

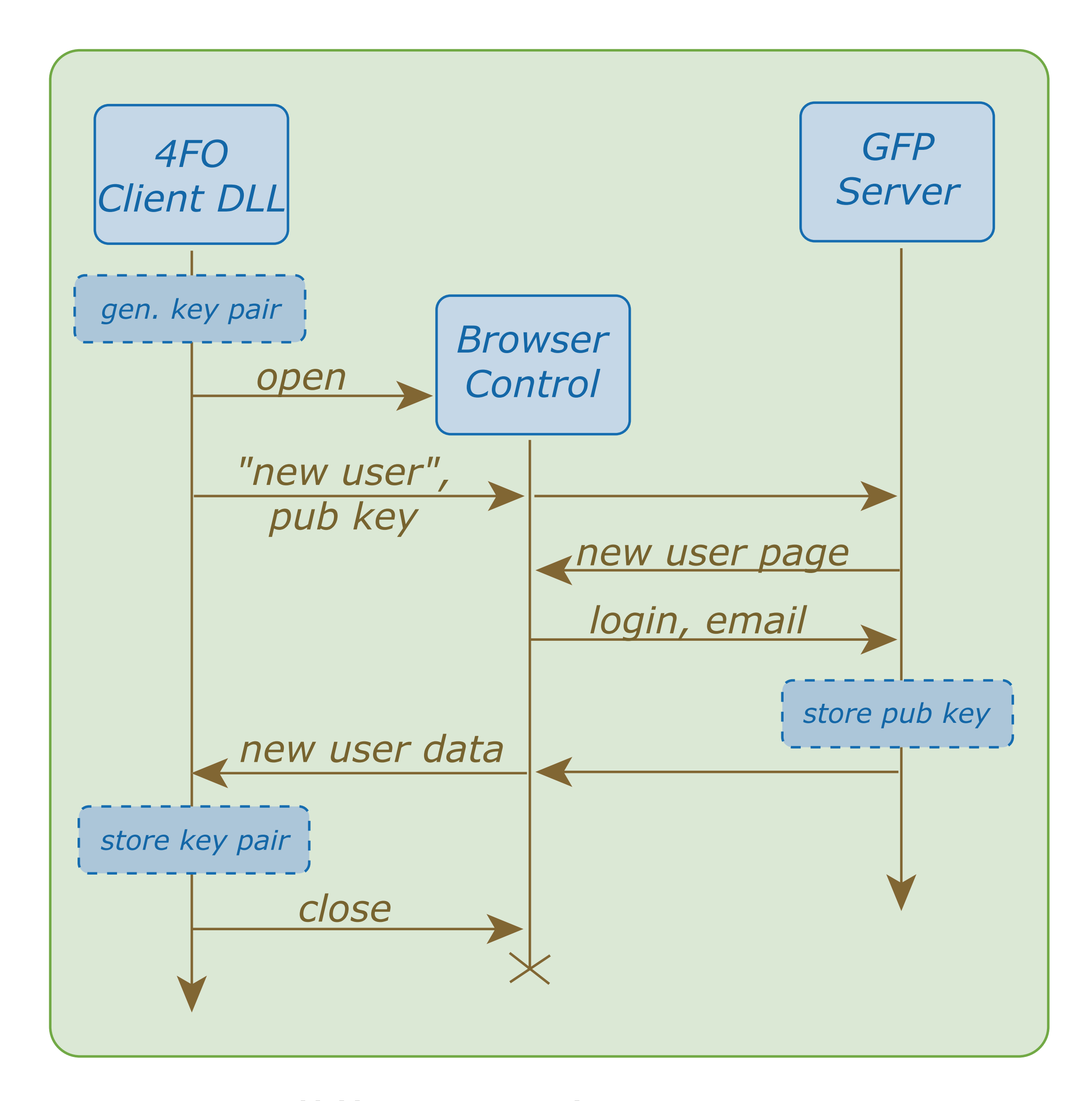


Abbildung 4.3: Sequenzdiagramm: „New User“

Wenn sich ein Kunde dazu entscheidet, den Server zu kontaktieren, um ein Feature herunterzuladen, werden im ersten Schritt auf dem Client zwei RSA-Schlüsselpaare generiert (1 Paar zum Ver- und Entschlüsseln und 1 Paar zum Signieren). Der private Schlüssel wird nochmals symmetrisch mit einigen nicht näher spezifizierten Hardwareeigenschaften verschlüsselt. Angelehnt an die Produktaktivierung von Microsoft sind die bezahlten und her-

untergeladen Features dadurch an den Rechner gebunden. Auf einem anderen Computer ließe sich der Schlüssel nicht wieder in seine ursprüngliche Form zurückverwandeln und wäre somit unbrauchbar. Das Web-Browser-Control wird geöffnet und der Client schickt eine Anfrage an den Server, die die ID des Spieles, den öffentlichen Schlüssel und den „new user" Modus enthält. Die Parameter sind zur Sicherheit mit dem privaten Schlüssel des Kunden signiert, um Man-in-the-Middle-Attacken zu verhindern. Wenn die Signatur mit dessen öffentlichen Schlüssel verifiziert werden konnte, antwortet der Server mit einem Login-Form. Der Benutzer wird nun aufgefordert, einen Loginnamen zu wählen sowie seine E-Mail-Adresse und andere optionale Daten anzugeben. Bestehen diese Daten einen Integrationscheck, wird ein generiertes Passwort, das jederzeit geändert werden kann, an die angegebene E-Mail-Adresse geschickt. Der öffentliche Schlüssel wird mit dem Loginnamen in einer Datenbank gespeichert. Als Letztes zeichnet sich das Spiel (nicht die DLL) für die Speicherung des generierten Schlüsselpaares verantwortlich.

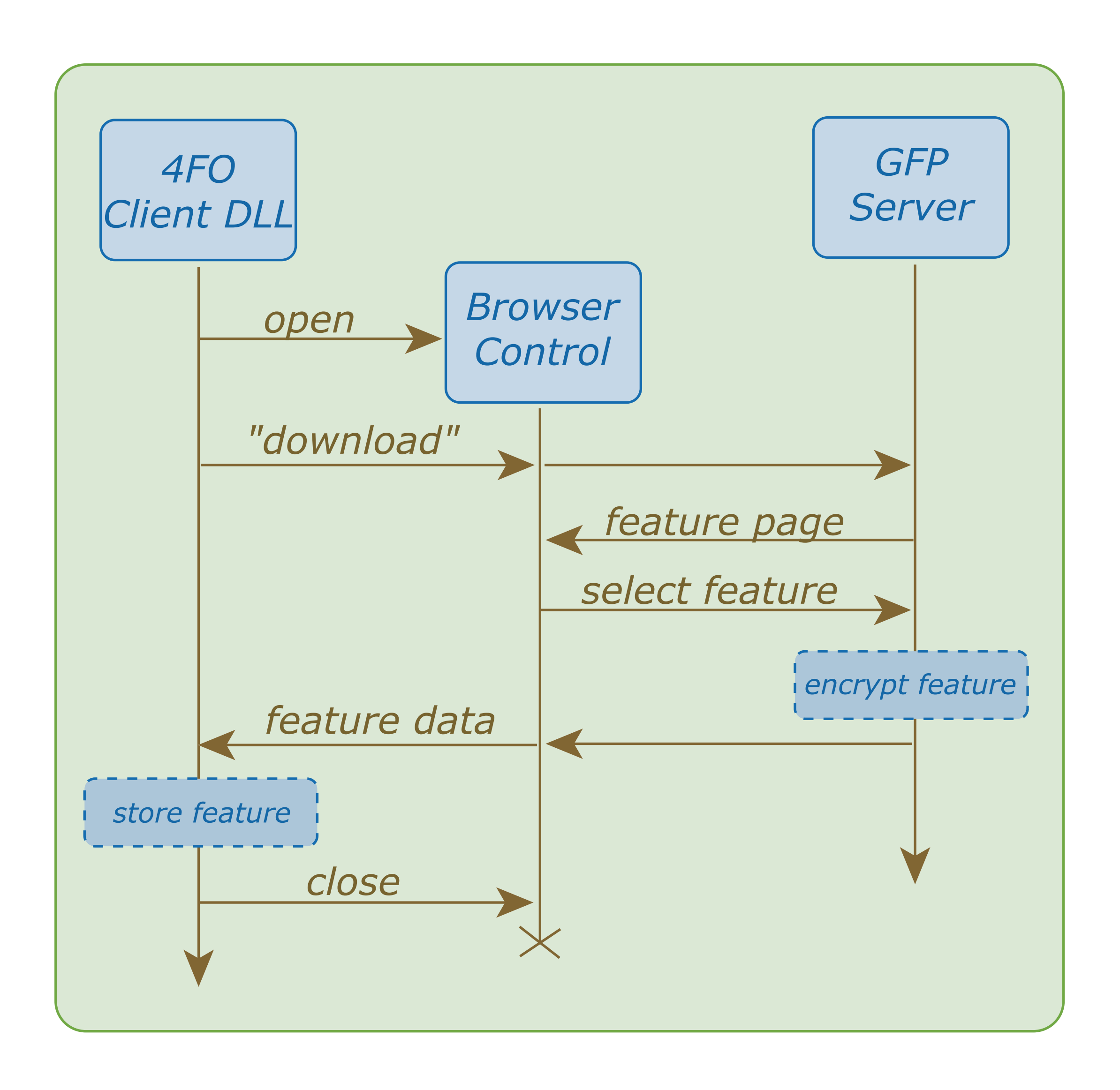


Abbildung 4.4: Sequenzdiagramm: „Download"

Die Abläufe beim Download eines Features sind in *Abbildung 4.4* dargestellt. Wenn der Benutzer im Spiel den Menüpunkt zum Herunterladen neuer Features anwählt, wird das bereits bekannte Web-Browser-Control aufgerufen. Diesmal werden Loginname, ID des betreffenden Spieles sowie der Kontext an den Server übermittelt. Auch hier sind die übertragenen Parameter nach dem selben Verfahren, wie oben aufgezeigt, signiert. Wird die Signatur erfolgreich verifiziert, erfolgt eine Darstellung der beziehbaren Features abhängig vom übergebenen Kontext. Nach Auswahl und Bezahlung durch den Kunden werden die gewählten Features einzeln in ein Archiv gepackt und mit dem Rijndael Algorithmus verschlüsselt. Der Download erfolgt mit Hilfe eines eingebauten ActiveX-Controls. Das so erstellte Archiv wird auf der Festplatte gespeichert und bei Bedarf durch den 4FO-Client im Arbeitsspeicher entschlüsselt. Als letzte Aktion wird das WebBrowser-Control geschlossen.

4.3 Probleme

Im letzten Abschnitt habe ich dargelegt, wie die Game-Feature-Plattform funktioniert und wie sie benutzt werden kann. Nun sollen einige Schwächen und Nachteile, die sich im Laufe der Zeit ergeben haben, aufgezeigt werden. Diese sollen als Ausgangspunkt für die Weiterentwicklung und damit den praktischen Teil dieser Diplomarbeit dienen.

Als ein großer Schwachpunkt stellte sich die Komplexität des Systems heraus. Auf der Client-Seite zum Beispiel gibt es für jeden Windows-Benutzer jeweils ein Login auf der Game-Feature-Plattform. Sollten Vater und Sohn an einem Rechner spielen, so werden sie als zwei unterschiedliche Kunden der GFP behandelt und müssen auch die Features separat beziehen. Des weiteren stellte sich die Nutzerverwaltung auf dem Client als zu aufwendig heraus. Dadurch erhöhte sich nicht nur die Schwierigkeit, sondern auch die Komplexität für die Spieleentwickler bei der Implementation. Damit einher geht der hohe Aufwand, den 4FO-Client in das Spiel zu integrieren. Ein Aufwand, den viele Entwickler scheuen.

Auf der Server-Seite zeigten sich ähnliche Schwierigkeiten. Der Game-Feature-Shop mit dem integriertem Archiv-Builder ist nicht modular aufgebaut. Bei der 4FriendsOnly.com AG ist man bestrebt, sich auf die Technologie des Archiv-Builder zu konzentrieren und vom Hosting des Game-Feature-Shops Abstand zu nehmen. Eine Trennung der Komponenten ist aber beim aktuellen Stand der Struktur kaum realisierbar.

Alles in Allem ist die Game-Feature-Plattform zu komplex angelegt. Oftmals würde eine einfache Freischaltung reichen, anstatt des komplizierten Feature-Downloads. Auch führt das Laden des Features aus dem Arbeitsspeicher zu Unbehagen unter den Entwicklern. Da die Probleme und Schwachstellen erkannt sind, gilt es nun sie zu lösen. Darum kümmern wir uns in den nächsten beiden Kapiteln.

Kapitel 5

Konzeption

Dieses Kapitel beschäftigt sich mit dem Entwurf der Server-Komponenten für die neue Game-Feature-Plattform. Da es sich hierbei um eine überarbeitete Version der bereits existierenden GFP handeln soll, habe ich sie *Game-Feature-Plattform Lite* genannt. Im nächsten Kapitel wird deren Implementation behandelt.

5.1 Zielsetzung

Wir haben im letzten Kapitel etwas über die bestehende Game-Feature-Plattform und im *Kapitel 4.3 (S.40)* etwas über deren Probleme erfahren. Aufgabe dieser Diplomarbeit ist es, die Schwachstellen auf der Server-Seite auszubessern. Das Grundprinzip der GFP (damit ist der Verkauf von einzelnen Komponenten gemeint) soll dabei erhalten bleiben. Es hat sich bewährt und führte zu einer guten Resonanz bei den Kunden.

Die Shop-Funktion soll nicht mehr die Hauptaufgabe der 4FriendsOnly.com Internet Technologies AG sein. Dort ist man bestrebt, sich auf den Archiv-Builder als Dienstleistung zu konzentrieren. Aus diesem Grund soll der monolithische Block des Game-Feature-Shops aufgelöst und der Archiv-Builder entkoppelt werden. Des Weiteren soll eine „Proof-Of-Concept"-Applikation des Game-Feature-Shops beweisen, dass die hier vorgestellte Lösung realisierbar ist. Es wäre denkbar, eine solche Anwendung als Referenzapplikation an interessierte Provider zu liefern, um eine mögliche Implementation aufzuzeigen. Wie der Provider letztendlich seinen Shop realisiert (ob in der hier dargestellten Programmiersprache PHP oder einer beliebig anderen) ist ihm überlassen. Einzige Bedingung ist, dass der Zugriff auf einen Web-Service möglich ist.

5.2 Aufbau

Ziel soll es sein, den Archiv-Builder vom Game-Feature-Shop zu trennen. Es muss also ein verteiltes System geschaffen werden.

Auf der einen Seite haben wir den Client. Dieser stellt nach wie vor einen Teil eines für die GFP modifizierten Programms dar. Der Client ist in der Lage, mit dem *Game-Feature-Shop Lite* zu kommunizieren. Wie genau, werden wir im nächsten *Kapitel 5.3 (S.45)* näher betrachten.

Auf der anderen Seite steht der *Game-Feature-Shop Lite*. Er wird das Portal für den Kunden sein. Hier wählt er die Features aus, die installiert werden sollen. Aus der Sicht des Kunden ist der Shop ein Server-System und sein einziger Kommunikationspartner. Um die Features aber zu kodieren, schlüpft der Shop in die Rolle eines Clients und kontaktiert einen dedizierten Server, der das Kodieren von Features übernimmt.

Für die Realisierung des entfernten Archiv-Builders bieten sich verschiedene Konzepte für verteilte Systeme an, die hier kurz genannt werden sollen. Als fortführende Literatur eignet sich „Verteilte Systeme“[OES01] von Andrew S. Tanenbaum.

- Das **Distributed Computing Environment (DCE)** ist eine Software-Technologie für das Erstellen und Verwalten von verteilten Systemen, die zum Industrie-Standard erhoben wurde. Es wurde von der Open Software Foundation (OSF) entwickelt, indem die Mitgliedsfirmen einige Software-Technologien beisteuerten und sich auf eine gemeinsame, herstellerunabhängige Lösung einigten. DCE wird typischer Weise in großen Netzwerken genutzt, die aus unterschiedlichen Servern bestehen und örtlich verteilt sind. Durch die Nutzung des Client/Server-Prinzips werden Anwendungsnutzer befähigt, Daten und Applikationen auf entfernten Rechnern zu nutzen. DCE bietet dabei eine Abstraktionsschicht, so dass Anwendungsentwickler sich keine Gedanken darüber machen müssen, wo die Programme laufen bzw. wo sich bestimmte Daten befinden. DCE unterstützt die prozedurale Programmierung verteilter Anwendungen. In der Interface-Beschreibungssprache von DCE werden Datentypen und Schnittstellen globaler Prozeduren und Funktionen definiert.

- Das **Distributed Component Object Model (DCOM)** besteht aus einer Reihe von Konzepten und Programmschnittstellen der Firma Microsoft, mit denen Client-Applikationen Services von Server-Objekten auf anderen Rechnern in einem Netzwerk nutzen können. DCOM basiert dabei auf Microsofts *Component Object Model (COM)*, welches die Kommunikation von Clients und Servern mit einer Reihe von Schnittstellen auf einem lokalen Rechner ermöglicht. *COM* wurde um eine verteilte Komponente erweitert und *Distributed COM* (oder *DCOM*) genannt. Es ist *CORBA* nicht ganz unähnlich und Microsofts Eigenentwicklung zur Realisierung von verteilten Systemen in der Windows-Welt.

- Javas **Remote Method Invocation (RMI)** ist eine Möglichkeit, die es Java-Entwickler ermöglicht, objektorientierte Anwendungen zu schreiben, in denen Objekte auf verschiedenen Rechnern über ein verteiltes System miteinandern interagieren können. RMI ist damit die Java-Version des *Remote Procedure Calls (RPC)* mit dem Unterschied, dass hier komplette Objekte mitgesendet werden können. RMI ist ein gut durchdachtes Konzept zur Implementierung eines verteilten Systems. Durch die Sprache Java hat man zwar eine Plattformunabhängigkeit gegeben, jedoch ist man an beiden Endpunkten der Kommunikation an diese Programmiersprache gebunden.

- Die **Common Object Request Broker Architecture (CORBA)** ist eine Architektur und Spezifikation zum Erstellen, Veröffentlichen und Verwalten von verteilten Objekten, welche die Interoperabilität von Anwendungen in einer verteilten Umgebung zum Ziel hat. Es erlaubt Programmen, die sich an unterschiedlichen Orten befinden und von unterschiedlichen Herstellern entwickelt worden sind, in einem Netzwerk zu kommunizieren. *CORBA* wurde von einem Hersteller-Konsortium, unter der Aufsicht der *Object Management Group (OMG)* entwickelt, dem mittlerweile mehr als 500 Firmen angehören. Das zugrunde liegende Objekt-Modell trennt zwischen Schnittstellenbeschreibung und Implementation. Es soll dadurch eine Kapselung erreicht werden, die unkontrollierbare Abhängigkeiten mit anderen Objekten vermeidet.

- Der **Web-Service** ist ein Software-System oder eine Anbindung an ein solches, das eindeutig über einen *Uniform Resource Identifier (URI)* angesprochen werden kann. Sie sind für die reine Maschine-Maschine-Kommunikation zwischen Software-Systemen gedacht, deren Bindungen lose sind. Es muss möglich sein, dass beide Kommunikationspartner sich zur Laufzeit finden (Just-in-time-Integration). Dazu werden die Schnittstellen öffentlich zugänglich gemacht und in einem XML-Datenformat beschrieben. Den ausgetauschten Daten liegt ebenso XML zugrunde. Dabei setzt die Kommunikation auf bestehende Internetprotokolle, wie HTTP oder SMTP.

Da es zwar eine Referenzimplementation zu dem *Game-Feature-Shop* geben wird, dieser jedoch auch durch Provider ersetzbar sein soll, fallen das *Distributed Component Object Model* und Javas *Remote Method Invocation* wegen ihrer Eingrenzung auf bestimmte Betriebsysteme oder Programmiersprachen heraus. Auch das *Distributed Computing Environment* steht wegen seiner eingeschränkten Möglichkeiten nicht zur Wahl. *CORBA* ist 'nur' eine Spezifikation. Es stehen zwar viele Referenzimplementationen zur Verfügung aber gerade deshalb kommt es zu vielen verschiedenen Systemen, die teilweise nicht kompatibel zueinander sind. Ein weiterer Nachteil ist, dass *CORBA* ein proprietäres Netzwerkprotokoll benutzt, welches durch viele Firmenfirewalls nicht durchgelassen wird. Aus den hier genannten Gründen habe ich mich dazu entschieden, Web-Services für den Archiv-Builder zu benutzen. Die Erstellung der notwendigen Dateien und die Implementation wird im *Kapitel 6.2.1 (S.54)* und folgende gezeigt.

In der gewählten Konstellation ist es der 4FriendsOnly.com Internet Technologies AG möglich, die Funktion des Archiv-Builders als Dienstleistung für Spiele-Provider zu verkaufen. Sie wenden sich mit Hilfe des Web-Service, der auf Basis der mitgezählten Aufrufe bezahlt werden kann, an den Archiv-Builder, welcher wiederum die Features für die Game-Feature-Plattform verschlüsselt. Ein Spiele-Provider ist für die tatsächliche Implementation des Shops selber verantwortlich. Eine im Rahmen dieser Diplomarbeit erstellte Referenzapplikation soll ihm aber eine gewisse Vorstellung geben, welche Möglichkeiten bestehen und wie der Web-Service mittels PHP aufgerufen werden kann.

5.3 Ablauf

Wir haben gesehen, wie die *Game-Feature-Plattform Lite* aufgebaut ist. Nun soll anhand eines weiteren Sequenzdiagrammes deren Funktion und Kommunikation zwischen den einzelnen Partnern dargestellt werden.

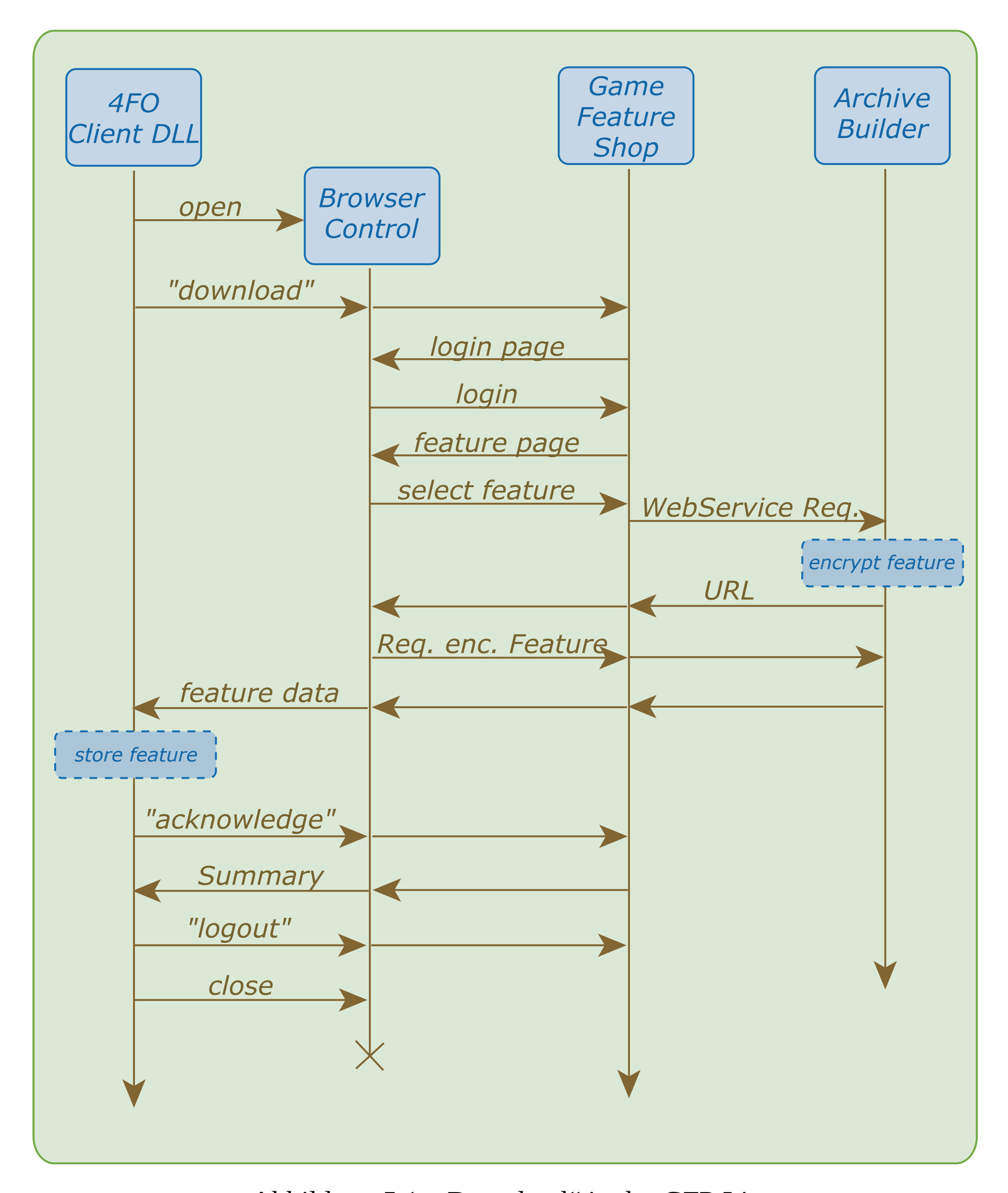


Abbildung 5.1: „Download“ in der GFP Lite

Abbildung 5.1 zeigt das Sequenzdiagramm für das Herunterladen eines neuen Features. Hat sich der Benutzer des Clients zu einer Erweiterung seines Programms entschieden, wird das Browser-Control geöffnet und die Parameter, die in *Tabelle* 6.3 aufgelistet sind, werden an den *Game-Feature-Shop Lite* übermittelt. Der User wird nun aufgefordert, sein Login einzugeben. Das ist für die eindeutige Identifizierung im Shop notwendig, da die Features auch bezahlt werden müssen. Es wären aber auch Szenarien vorstellbar, in denen kein Login

benötigt wird. Technisch gesehen ist es nicht erforderlich. Wurde das Login erkannt, erfolgt eine Präsentierung der installierbaren Features abhängig vom Kontext. Es wäre zum Beispiel vorstellbar, dass ausschließlich weiterführende Level als Features angeboten werden, wenn der Kunde das letzte freie Level durchgespielt hat und nun weitermachen möchte.

Wie bereits angedeutet ist es neben einer einfachen Freischaltung auch möglich, gleich mehrere Features – also Level oder Karten – auf einmal herunterzuladen. Hat sich der Kunde entschieden, werden die gewünschten Features durch den Archiv-Builder mittels eines Web-Service-Aufruf verschlüsselt. Dabei werden die in *Tabelle 5.1* aufgelisteten Parameter übertragen.

Name	Bedeutung
gameID	Eine Zahl, um Spiel und Version genau zu identifizieren.
originalFileName	Der Dateiname des Features.
featureURL	Webadresse, von der das unverschlüsselte Feature geladen werden kann.
customerPublicKey	Base64-codierter Public Key Modulus des Kunden.

Tabelle 5.1: *An den Web-Service übermittelte Parameter*

Die unverschlüsselten Features liegen immer bei dem Shop-Betreiber. Damit der Archiv-Builder das Feature auch verschlüsseln kann, muss ihm die unkodierte Version zur Verfügung stehen. Diese in dem initialen Web-Service-Aufruf mit zu übergeben, wäre auf Grund des mitunter binären Charakters der Datei unangebracht. Aus diesem Grund habe ich mich zu folgender Lösung entschieden. Dem Web-Service wird eine URL zu dem unverschlüsselten Feature übergeben, von der es geladen werden kann. Dabei ist von dem Shop-Betreiber sicherzustellen, dass die Datei in einem geschützten Bereich liegt, der 'normalen' Benutzern nicht zugänglich ist. Man kann dies erreichen, indem Zugriffe auf diesem Bereich nur von einer IP – nämlich der des Archiv-Builders – erlaubt sind. Die Game-ID sowie der Dateiname des Features werden für die Verschlüsselung benötigt. Das Archiv wird unter einem temporären Dateinamen auf dem Server des Archiv-Builders abgelegt und die URL zu dieser Datei als Ergebnis des Web-Services an den aufrufenden *Game-Feature-Shop-Lite* zurückgeliefert. Dieser baut im Anschluss eine dynamische Seite mit besagter URL zusammen, die ein ActiveX-Control enthält. Einmal auf dem Client angezeigt, beginnt das Control den Download des verschlüsselten Features von dem Rechner des Archiv-Builders. Wird das Herunterladen erfolgreich abgeschlossen, meldet der Client dies dem Shop durch eine *Acknowledge*-Nachricht. Das ist das Zeichen, um dem Kunden eine Seite mit dem Erhalt des oder der Features zu präsentieren. Bestätigt er den Vorgang, wird das Browser-Control geschlossen.

5.4 Weiterführende Überlegungen

Abschließend sollen noch einige Betrachtungen zur Sicherheit, Fair-Use und Transaktionssicherheit folgen. Sie sollen einen Überblick geben, worauf man bei dem Einsatz der Game-Feature-Plattform noch alles zu achten hat.

5.4.1 Sicherheit

Wir sind uns alle einig, wenn ich sage, dass es eine hundertprozentige Sicherheit nie geben wird. Ein erklärtes Ziel ist es aber, die „Hobby-Hacker“ abzuschrecken. Dabei muss der Aufwand zur Realisierung des Schutzes im Verhältnis zum Wert des Produktes stehen. Rein theoretisch gibt es mehrere Angriffspunkte am System der Game-Feature-Plattform. Diese sollen im Folgenden vorgestellt werden.

Als erstes wäre da der Client (das Spiel) selber. Da im Prinzip jeder ein Spiel aus dem Internet herunterladen kann, welches an die GFP angepasst wurde, besteht hier wohl das größte Potential eines Angriffs. Leute mit genügend krimineller Energie und vor allem dem nötigen Wissen könnten ein solches Programm dekompilieren. Ein jeder Kopierschutz geht mit einer einfachen Ja/Nein-Entscheidung im Quellcode einher. Selbst die kompliziertesten Vorkehrungen gehen irgendwann auf eine einfache Entscheidung zurück. Genau hier setzt ein Hacker an. Nachdem besagte Stelle gefunden wurde, wird die Ja/Nein-Entscheidung einfach umgedreht. Somit ist das Programm freigeschaltet. Diese Vorgehensweise lässt sich aber nur für eine Freischaltungen umsetzen, da sie besagte Ja/Nein-Entscheidungen enthalten. Neben einer Freischaltung kann sich in einem über die Game-Feature-Plattform ausgelieferten Feature auch Content befinden. Man könnte sich hier einen Programmabschnitt vorstellen, der „echte Daten“ zur Abarbeitung beiträgt – sich also nicht nur auf eine einfache Ja/Nein-Entscheidung beschränkt. Wenn dies genutzt würde, wäre die Gefahr eines Angriffs an diesem Punkt quasi nicht existent.

Ebenso wäre die Gefahr eines Angriffs auf den Shop eines Providers als gering einzustufen. Da wir davon ausgehen können, dass mit dem Verkauf von Features Gewinn erzielt werden soll, sollten alle denkbaren Vorkehrungen zum Schutz vor einem Einbruch getroffen werden. Selbst wenn es jemandem gelingen sollte, an die unverschlüsselten Features zu gelangen, muss immer noch sichergestellt werden, dass diese durch den Web-Service mit dem richtigen Kundenschlüssel verschlüsselt werden. Ansonsten wird der 4FO-Client diese Datei als ungültig erkennen.

Als hoch würde ich das Potential eines Angriffs auf den Web-Service selber einschätzen. Dabei ist weniger der Diebstahl von Daten als eine groß angelegte *Distributed Denial of Service (DDOS)*-Attacke gemeint. Dies würde zur Nicht-Verfügbarkeit des Web-Services und damit des ganzen Systems führen. Dies würde nicht nur einen sondern alle Provider betreffen. Ein solcher Angriff könnte zu einem erheblichen Imageschaden führen, wenn das

Herunterladen von Features wiederholt nicht möglich ist. Hier wäre ausreichende Redundanz empfehlenswert.

Neben den Angriffen auf die einzelnen Komponenten selbst wäre noch der Netzwerkverkehr zu betrachten. Meiner Einschätzung nach ist die Gefahr einer *Man-in-the-Middle*-Attacke – also das Abhören des Netzwerkverkehrs – als gering bis nicht existent einzustufen. Selbst wenn es jemandem gelingen würde, den Netzwerkverkehr zwischen Client und Shop oder zwischen Shop und Web-Service abzufangen, könnte er mit den gewonnen Daten vermutlich nicht viel anfangen. Sie sind entweder so unwichtig oder so gut verschlüsselt, dass es Jahrzehnte zur Dekodierung brauchen würde.

Ein letzter Angriffspunkt wurde aber noch nicht betrachtet. Dieser ist auf den ersten Blick auch nicht leicht zu erkennen. Die Verbindung zwischen dem Spiel und dem 4FO-Client (der DLL) kann ebenfalls einer *Man-In-The-Middle*-Attacke unterliegen. Zuerst wäre es notwendig, ein „Spiel“ zu entwickeln, dessen einzige Aufgabe es ist, die gewünschten Features vom *Game-Feature-Shop Lite* zu beziehen und diese im unverschlüsselten Zustand auf der Festplatte zu speichern. Aufbauend auf dieser Tatsache müsste nun eine neue DLL programmiert werden, die sich nach außen hin wie die Originalversion verhält, allerdings auf eine Internetverbindung verzichtet und als Featurequelle die unkodierten Dateien aus dem letzten Schritt benutzt. Die so veränderte DLL kann nun mit dem Originalspiel verteilt werden und würde dieses freischalten. Auch hier stellt sich wieder die Frage nach dem Kosten-Nutzen-Verhältnis. Klar ist, dass dies nur mit tiefgreifenden Kenntnissen geschehen kann. Es ist sicher nicht unmöglich, jedoch auch nicht sehr wahrscheinlich – zumal es nur ein Spiel, das auf die GFP aufbaut, freischalten würde. Als eventuelle Lösung wäre das statische Einbinden der DLL in die Anwendung vorstellbar. Dies würde aber wiederum zu Mehraufwand bei den Entwicklern und anderen Schwierigkeiten bei der Entwicklung führen.

Es ist auch zu bedenken, dass die Schlüssel, die zu Beginn generiert werden, in Verbindung mit ihrer Hardwarebindung der einzige Sicherheitsanker sind. Wenn jemand diese Schlüssel kopiert, kann er auch die dazugehörigen Features kopieren – unter der Voraussetzung es gelingt ihm, die symmetrische Verschlüsselung zu knacken. Eine manipulierte DLL, die immer die gleichen Schlüssel generiert, wäre wohl ein sehr hohes Sicherheitsrisiko. Hier müsste der Shop-Provider eine Logik implementieren, die es ihm gestattet zu überprüfen, ob nicht zwei verschiedene Kunden ihm die selben Schlüssel schicken.

5.4.2 Fairer Umgang mit Nutzern

Da es ein Ziel der Game-Feature-Plattform ist, ein faires Kopierschutzsystem anzubieten, soll im Folgenden näher auf die Thematik eingegangen werden.

Was ist überhaupt fair? Das Lexikon definiert fair als *„a) anständig, ehrlich, gerecht; b) den [Spiel]regeln entsprechend, sie beachtend, kameradschaftlich (Sport)“*. Was bedeutet also ein gerechter Kopierschutz für den einzelnen Nutzer? Was bedeutet er für den Spiele-

entwickler?

Im *Kapitel 2.2.3 (S.12)* über die Shareware wurde gesagt, dass ein Autor einer Shareware-Software keine Kontrolle darüber hat, wer und vor allem wie lange das von ihm entwickelte Produkt eingesetzt wird. Sätze wie: „Seien Sie fair gegenüber dem Autor, indem Sie die Registrierungsgebühr bezahlen“ prangen deshalb oft auf Erinnerungsdialogen. Fair gegenüber dem Autor bedeutet also, dass dieser das ihm zustehende Geld für seine Arbeit bekommt. Doch sich dem Kunden gegenüber fair zu erweisen stellt sich als wesentlich schwieriger heraus. Da jeder eine andere Auffassung von „fair“ hat, entschloss man sich, die Umsetzung dem Provider zu überlassen. Dieser hat die Möglichkeit durch die Implementation von Regeln seine Vorstellungen vom „Fair-Use“ zu verwirklichen. Hier nun einige Betrachtungen, was zukünftige Shop-Betreiber beachten sollten:

Die Features, die über die Game-Feature-Plattform ausgeliefert werden, sind mit dem Schlüssel des Kunden verschlüsselt. Dieser wiederum ist symmetrisch mit Hardwareparametern verschlüsselt. Auf einem Rechner mit anderer Hardware wäre er nicht in seinen Ursprungszustand zurückzuführen. Somit sind die Features an einen Rechner gebunden. Es gibt aber auch Nutzer, die die gekauften Features auf einem zweiten Computer (ein Laptop vielleicht) einsetzen wollen. Es wäre kulant – und somit auch fair – wenn der Shop-Provider dem Kunden eine gewisse Anzahl von Installation gestattet. Es wäre denkbar, dass es jedem Kunden erlaubt ist, zum Beispiel drei verschiedene Schlüsselpaare (für drei verschieden Rechner) zu generieren und diese auf dem Server speichern zu lassen. Ein Nachteil, den der Kunde in Kauf nehmen müsste, ist, dass er für jeden seiner Computer die Features erneut herunterladen müsste. Für kleine Dateien ist das sicherlich unproblematisch.

Große Dateien können Quelle einer anderen Schwierigkeit sein. Angenommen, einer der Hardwareparameter, die zum Verschlüsseln des Keys benötigt werden, ist die CPU in einem Computer und der Kunde rüstet den Prozessor auf. In diesem Fall werden mit einem Schlag alle Features „ungültig“, da sie sich nicht mehr entschlüsseln lassen. In einem solchen Szenario wäre es denkbar, dass man die Schlüssel redundant abspeichert. Möchte man vier Hardwareparameter einbauen, so generiert man ein Schlüsselpaar, welches viermal kopiert wird. Die so entstandenen Duplikate werden jeweils mit drei der vier gewünschten Hardwareparameter verschlüsselt. Sollte einer der Komponenten ausgetauscht werden, ist noch immer einer der Schlüssel gültig, nämlich genau der, der mit den jeweils anderen drei Komponenten verschlüsselt wurde.

Weiterhin wäre es denkbar, dass die Festplatte eines Kunden zerstört wird und somit alle Daten (inkl. des privaten Schlüssels) verloren gehen. In einem solchen Fall sollte sich der Provider als so kulant zeigen und das erneute Generieren eines Schlüsselpaares erlauben. Dieses sollte auf dem Server gespeichert werden und einen eventuell vorhandenen alten Schlüssel überschreiben. Das Abspeichern hat den Grund, dass der Provider bei jeder Anfrage überprüfen kann, ob der Kunde noch seinen Schlüssel besitzt. Wird dieser zu oft geändert, könnte das ein Anzeichen dafür sein, dass das Login weitergegeben wurde. Ein eingebauter

Kulanz-Zähler könnte zum Beispiel nach 10-maligem Ändern den Account sperren. Daraufhin müsste der Kunde sich telefonisch bei dem Shop-Provider melden um seinen Account zu reaktivieren. Dieser Schritt wird von Raubkopierern gemieden, da sie in der Regel die „sichere" Anonymität nicht verlassen wollen.

Ein weitaus komplizierterer Punkt und hier ist viel „Treu und Glauben" im Spiel, ist die Deinstallation. Man stelle sich vor, dass die Game-Feature-Plattform für eine Software eingesetzt wird, die der Kunde im Geschäft bezahlt hat. Zur Freischaltung hat er einen Code bekommen, den er an Stelle eines Logins eingeben muss. Möchte dieser Kunde die Software nun veräußern, muss er in der Lage sein, das Produkt in der GFP zu „de-registrieren", d.h. der Code muss wieder als „frei" markiert werden. Dieser Vorgang besteht aus zwei Teilen. Da wäre die eigentliche Deinstallation auf dem Rechner des Kunden sowie die Meldung an den Server, dass die Deinstallation erfolgreich war. Wird die Deinstallation angestoßen, aber die abschließende Erfolgsnachricht nicht vom Server empfangen, kann nicht mit Sicherheit gesagt werden, ob die Kommunikation zwischen beiden Partnern nun mutwillig unterbrochen wurde oder ob die Meldung nur verloren ging oder ob die Deinstallation gar nicht erfolgreich war. Ein weiterer Schwachpunkt ist die Wiederherstellung mittels eines Backups. Ein Kunde hat sein Programm ordnungsgemäß deinstalliert und danach ein früheres Backup wieder eingespielt. Somit wäre die Software auf der Game-Feature-Plattform wieder als frei markiert und der Kunde könnte das Programm weiter benutzen (allerdings illegal). Hier kommt das eingangs erwähnte „Treu und Glauben" wieder ins Spiel. Ein solcher Fall sollte fairer Weise immer zu Lasten des Providers gehen, um Kunden nicht zu verärgern. Missbrauch kann über den vorhin erwähnten Kulanz-Zähler verhindert werden.

5.4.3 Transaktionssicherheit

Zum Schluss wäre noch zu klären, was passiert, wenn der Ablauf, der in *Kapitel 5.3 (S.45)* beschrieben wurde, so nicht stattfindet, sondern durch beabsichtigtes oder unbeabsichtigtes Eingreifen gestört wird.

Wenn man sich den Aufbau der Game-Feature-Plattform noch einmal vor Augen führt, kann man sich zwei kritische Punkte vorstellen, die zu Problemen führen können. Zum einen wäre da die Netzverbindung zwischen *Game-Feature-Shop Lite* und dem Web-Service. Sollte diese Strecke ausfallen, wäre das Generieren von Archiven nicht möglich. Als Lösung wäre vorstellbar, dass Features vorsorglich kodiert werden. Wenn der Kunde also seine Wahl getroffen hat (jedoch noch vor der Bezahlung), wird versucht, den Web-Service zu kontaktieren. Ist dies nicht möglich, kann dem Kunden eine Nachricht angezeigt werden, den Vorgang zu einem späteren Zeitpunkt zu wiederholen. Sollte das Feature jedoch kodiert worden sein, der Bezahlvorgang aber fehlschlagen oder nicht beendet werden, ist der einzige Nachteil, dass das verschlüsselte Feature auf dem Archiv-Builder liegt. Da die Datei einen temporären Namen haben sollte, ist das Herunterladen ohne die passende URL ausgeschlossen.

Der andere kritische Punkt ist die Kommunikation zwischen dem 4FO-Client und dem *Game-Feature-Shop Lite*. Solange die Bezahlung noch nicht abgeschlossen ist, kann der gesamte Vorgang jederzeit ohne Nachteile wiederholt werden. Kritisch wird es erst, wenn die Bezahlung erfolgt ist. Schließlich möchte der Kunde die Ware, die ihm zusteht. Da jedes Feature mit einem Schlüssel kodiert wird, der nur von genau diesem einen Kunden auf genau einem Rechner genutzt werden kann, sollte es keine Probleme bereiten, wenn das Feature so oft heruntergeladen werden kann, wie es benötigt wird. Für den Fall dass der Kunde die Datei also nach dem Bezahlvorgang nicht erhalten hat, weil die Internetverbindung unterbrochen worden ist, kann er es zu einem späteren Zeitpunkt noch einmal versuchen. Der Shop sollte erkennen, dass das angeforderte Feature von diesem Kunden bereits bezahlt wurde und es unentgeltlich noch einmal zum Download zur Verfügung stellen. In der mutwilligen Zurückhaltung der Bestätigungsmeldung des Clients an den Shop sehe ich keine Vorteile. Das Feature wurde bereits bezahlt und der Client hat die Datei erhalten. Diese Nachricht dient auch nur dazu, das Web-Control auf dem Client zu schließen.

Abschließend kann man sagen, dass die Game-Feature-Plattform Lite so konzipiert wurde, dass es zu keinen wesentlichen Nachteilen auf der Seite des Kunden oder des Providers kommen kann.

Kapitel 6

Implementierung

Dieses Kapitel behandelt den praktischen Teil der Diplomarbeit. In diesem Abschnitt wird auf die bei der Implementierung verwendeten Software und auf die Umsetzung der Konzeption eingegangen.

6.1 Software

Während der Implementierung wurde eine Reihe von Software und Tools benutzt, die in *Tabelle* 6.1 aufgeführt sind. Die Bezugsquellen und die Dokumentation kann über den ebenso aufgeführten Link in Erfahrung gebracht werden.

Name	**Beschreibung**	**Link**
UltraEdit v10.0	Texteditor zum Erstellen des PHP Codes	`http://www.ultraedit.com/`
IntelliJ IDEA v4.0	Entwicklungsumgebung für Java	`http://www.jetbrains.com/idea/`
Axis v1.1	SOAP-Implementierung und Toolkit	`http://ws.apache.org/axis/`
Tomcat v5.0.18	Servlet-Container für Axis	`http://jakarta.apache.org/tomcat/`
Java 2 SDK, SE v1.4.2_04	Java Virtual Machine	`http://java.sun.com`
Apache v2.0.50	Webserver	`http://www.apache.org/`
SOAP Package	Client/Server-Bibliothek für PHP	`http://pear.php.net/package/SOAP`

Tabelle 6.1: *Für die Implementierung benutzte Software*

6.2 Der Prototyp

Ein Prototyp ist ein vorläufiges oder temporäres Produkt, mit dem künftige Anwender ausgewählte Eigenschaften und Funktionen des zu entwickelnden endgültigen Systems erproben können. Dabei muss der Prototyp nicht alle Funktionen des endgültigen Produktes anbieten, sollte aber einfach erweiterbar und zu verändern sein. Prototypen können explorativ sein, d.h. zur Erforschung eines Sachverhaltes dienen. Sie können experimentell, d.h. zur Überprüfung der Machbarkeit oder Funktionsfähigkeit dienen, oder evolutionär sein, d.h. vorab ein Teilprodukt bereitstellen (Vgl. [OES01]).

6.2.1 Erstellung der WSDL-Beschreibung

Die *Web Service Description Language (WSDL)* ist ein Hilfsmittel zur strukturierten Beschreibung eines Web-Services. Sie definiert eine XML-Grammatik, die einerseits als Dokumentation aber auch als Skizze zum automatischen Aufbau von Kommunikationswegen dienen kann. Die Operationen und Nachrichten werden abstrakt beschrieben und an konkrete Netzwerkprotokolle und Nachrichtenformate gebunden.

Um die Definition einer in WSDL spezifizierten Schnittstelle zu verstehen, sind grundlegende XML-Kenntnisse notwendig. Das Prinzip aber lässt sich einfach skizzieren. Eine WSDL-Beschreibung ist hierarchisch aufgebaut und enthält fünf Abschnitte, die durch die XML-Elemente *<types>*, *<message>*, *<portType>*, *<binding>* und *<service>* gekennzeichnet sind. Die ersten drei Abschnitte (*<types>*, *<message>*, *<portType>*) nennt man den „abstrakten“ Teil, und die beiden letzten (*<binding>*, *<service>*) den „konkreten“ Teil einer WSDL-Beschreibung. Im abstrakten Teil werden unabhängig von einem Protokoll oder Service die Operationen und verwendeten Datentypen beschrieben. Im konkreten Teil wird beschrieben, über welche URI (Uniform Resource Identifier) und Protokolle ein Web-Service erreicht werden kann und wie die Daten serialisiert und codiert werden. Diese muss, wie in *Kapitel 6.3 (S.62)* beschrieben, angepasst werden.

Der Web-Service zum Erstellen des Archivs heißt **ArchiveService** und wird in der *service*-Sektion, die in *Listing 6.1* dargestellt ist, definiert.

```
<service name="ArchiveService">
    <port name="Archive" binding="tns:ArchiveSOAPBinding">
        <soap:address location="http://localhost:8080/axis/services/Archive"/>
    </port>
</service>
```

Listing 6.1: *WSDL: Service Definition*

In diesem Abschnitt des WSDL-Files ist außerdem die Adresse definiert, unter der der Web-Service zu erreichen ist. Durch Anhängen der Zeichenkette `?wsdl` an die URL erhält man eine vom Server generierte WSDL-Beschreibungsdatei. Dieses Verhalten ist besonders sinnvoll, wenn man gezwungen ist, einen unbekannten Web-Service anzusprechen. Ohne

zusätzliche Dokumentation des Entwicklers hat man alle nötigen Daten um den Web-Service erfolgreich aufrufen zu können.

Ein Web-Service hat mindestens einen kann aber auch mehrere *Ports* enthalten. Diese sind mit dem Interface-Konzept von Java vergleichbar. Sie stellen eine Sammlung von *Operationen* dar. Ein Port kann demnach auch mehrere Operationen (oder Funktionen, um den Vergleich mit Java wieder heranzuziehen) enthalten. Unser Web-Service besitzt nur einen Port mit einer Operation.

```
<portType name="Archive">
    <operation name="createArchive">
        <input message="tns:CreateArchiveRequest" />
        <output message="tns:CreateArchiveResponse" />
    </operation>
</portType>
```

Listing 6.2: *WSDL: PortType Definition*

Die Operation (*Listing 6.2*) heißt **createArchive**. Eine Operation kann maximal ein Set von Eingabeparametern (*Listing 6.3*) und ein Set von Ausgabeparametern (*Listing 6.4*) haben.

```
<message name="CreateArchiveRequest">
    <part name="gameID" type="xs:int" />
    <part name="originalFileName" type="xs:string" />
    <part name="featureURL" type="xs:string" />
    <part name="customerPublicKey" type="xs:string" />
</message>
```

Listing 6.3: *WSDL: Message Definition für Eingabeparameter*

Beide Parameterarten sind als *message* (engl. für Nachricht) definiert. Der Grund für die Einführung einer extra Hierarchiestufe für die Parameter ist auf den ersten Blick nicht sofort ersichtlich – läßt sich aber mit dem Konzept der Wiederverwendung erklären. Man stelle sich vor, dass unser Web-Service eine zweite Operation enthält, die vielleicht die gleichen Eingabeparamaeter benötigt und es seien nicht vier sondern zehn an der Zahl. Hier fällt die Wiederholung der Zeile `<input message="tns:CreateArchiveRequest" />` deutlich leichter als alle 10 Parameter nochmals aufzuzählen. Selbes gilt natürlich für die Ausgabeparameter.

```
<message name="CreateArchiveResponse">
    <part name="downloadURL" type="xs:string" />
</message>
```

Listing 6.4: *WSDL: Message Definition für Ausgabe*

Da Ein- und Ausgabeparameter nur primitive Datentypen verwenden, können wir gänzlich auf die Definition eigener Typen verzichten. Dies erhöht nicht nur die Lesbarkeit sondern reduziert auch die Größe das WSDL-Files.

Zum Schluss fehlt noch das eine wichtige Element, was unseren Web-Service mit seinen Operationen zusammenführt – das *Binding*. Hier wird das Transportprotokoll für unseren Web-Service bestimmt. *Listing* 6.5 zeigt den betreffenden Ausschnitt aus unserem WSDL-File. Üblicherweise greift man hier auf *Simple Object Access Protocol* (SOAP) zurück. Jedoch ist auch jede andere Form des Transports vorstellbar.

```
<binding name="ArchiveSOAPBinding" type="tns:Archive">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />

    <operation name="createArchive">
        <soap:operation soapAction="urn:ArchiveBuilder.Web-Services.gfp.ffo.com" />
        <input>
            <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="urn:ArchiveBuilder.Web-Services.gfp.ffo.com" use="encoded"
                 />
        </input>
        <output>
            <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="urn:ArchiveBuilder.Web-Services.gfp.ffo.com" use="encoded"
                 />
        </output>
    </operation>
</binding>
```

Listing 6.5: *WSDL: Binding Definition*

Das vollständige File befindet sich auf der beiliegenden CD im Verzeichnis *webservice/wsdl/ArchiveBuilder.wsdl*

6.2.2 Generierung der Java-Klassen

Bei der Implementierung der Java-Klassen wird man durch das AXIS Toolkit unterstützt. Anhand der im letzten Abschnitt erstellten WSDL-Beschreibung lassen sich Stubs (Code für die Client-Seite) und Skeletons (Code für die Server-Seite) generieren. Für die Game-Feature-Plattform wollen wir nur einen Service anbieten, nämlich das Erstellen eines kodierten Archivs. Standardmäßig werden aber nur die Stubs generiert. *Listing* 6.6 zeigt, wie man mit Hilfe des WSDL2Java Tools, welches Bestandteil des AXIS Toolkits ist, die benötigten Dateien erstellen kann. Die Parameter `--server-side` und `--skeletonDeploy true` erzwingen die Erstellung des für uns wichtigen server-seitigen Codes.

```
# java org.apache.axis.wsdl.WSDL2Java --server-side
  --skeletonDeploy true ArchiveBuilder.wsdl
```

Listing 6.6: *Erstellen des Java Skeletons*

Die generierten Dateien werden in das Verzeichnis *com/ffo/gfp/Web-Services/ArchiveBuilder* geschrieben. Sie landen dort, weil der Target-Name-Space (TNS) des WSDL-Files auf die Java-Paket-Struktur gemappt wird. Durch das Tool WSDL2Java wurden folgende Java-Klassen erstellt, die in *Tabelle* 6.2 aufgelistet sind. Diese können auch auf der beigelegten CD im Verzeichnis *webservice/generierte_dateien* eingesehen werden.

WSDL Klausel	Generierte Klasse
Für jedes PortType	Ein Java Interface, um auf die Operationen des Web-Services zugreifen zu können.
Für jedes Binding	Eine Java Stub- und Skeleton-Klasse, um die Methodenaufrufe des Web-Services in SOAP-Request umzuwandeln. Damit verhält sich der Web-Service wie ein lokales Objekt. Eine Templateklasse für die eigentliche Implementation. Hier erfolgt die im nächsten Abschnitt beschriebene Implementation
Für jeden Service	Ein Interface und eine Implementation für den Service-Locator. Diese Klasse wird benötigt, um den Web-Service anzusprechen, wenn der generierte Code als Client benutzt wird.
Für alle Services	Eine deploy.wsdd und undeploy.wsdd Datei, die zur automatisierten Installation und Deinstallation in Tomcat benötigt werden.

Tabelle 6.2: *Generierte Java-Klassen*

6.2.3 Umsetzung des Web-Service

Nun, da wir alle benötigten Dateien haben, steht der eigentlichen Implementation des Web-Services nichts mehr im Wege.

Listing 6.7 zeigt die Implementation des Web-Service in Auszügen. Dabei wurden sämtliche Sourcecode-Kommentare entfernt, sowie auf den Exception-Handling-Code verzichtet, um die Sache an dieser Stelle nicht unnötig aufzublähen. Der vollständige Sourcecode ist wie gewohnt auf der beiliegenden CD im Verzeichnis *java/src/com/ffo/gfp/webservices/ArchiveBuilder* enthalten.

Der eigentliche Web-Service besteht nur aus einigen wenigen Zeilen Code und soll hier in Kürze erklärt werden:

- **Zeile 3:** Das Feature liegt in einem gesicherten Bereich, der nur durch den Web-Service selbst erreicht werden kann. Die URL dorthin wird dem Web-Service als Parameter übergeben. In dieser Zeile wird mittels der Javaklasse `java.net.URL` dieses in ein Byte-Array geladen.

- **Zeile 5-11:** Es wird eine Instanz (*arc*) des Objektes (*CArchives4fo*) erstellt, welches für das Verschlüsseln des Features verantwortlich ist. Diese Klasse ist Bestandteil der existierenden GFP. Neben dem Ver- und Entschlüsseln ist sie auch für das Zusammenführen von mehreren Features in ein Archiv verantwortlich. Im Anschluss werden noch einige dringend notwendige Eigenschaften gesetzt. Der öffentliche Schlüssel des Kunden wird für die Kodierung benötigt. Um die Transfermenge der Parameter so gering wie möglich zu halten, wird nur der Modulus des öffentlichen Schlüssels übertragen. Der Exponent ist mit dem Wert 65537 konstant.

```
public String createArchive(int gameID, String originalFileName, String featureURL,
    String customerPublicKey) throws RemoteException {

    byte[] data = this.loadFeatureFromHTTP(featureURL);

    CArchives4fo arc = new CArchives4fo();
    arc.setArchiveVersion(4);
    arc.setCustomerPubKey(customerPublicKey, Base64.encode(new BigInteger("65537").
        toByteArray()));
    arc.setGameID(gameID);
    arc.setGamePrivKey(GameDB.getInstance().getPrivKeyModulus(gameID), GameDB.
        getInstance().getPrivKeyExponent(gameID));
    arc.setCustomerName("");
    arc.setSessionID("");
    arc.addPlugin("1", data, originalFileName, 0, 0, 0, (data.length + 15) / 16, 0,
        0, "", "");

    EncodedFileLocation encodedFileLocation = new EncodedFileLocation(
        originalFileName, customerPublicKey);
    encodedFileLocation.putFile(arc.toByteArray());

    return encodedFileLocation.getURL();

}
```

Listing 6.7: *Auszug aus der Web-Service Implementation*

- **Zeile 12:** Das Feature, welches sich in dem Byte-Array *data* befindet, wird kodiert. Die übergebenen Parameter definieren Start, Blockgröße, Dateigröße, etc. für die Verschlüsselung.

- **Zeile 14-15:** Mit Hilfe von *EncodedFileLocation* wird das kodierte Feature auf die Festplatte geschrieben. Dabei wird ein MD5-Hash über den Dateinamen des Features sowie den öffentlichen Schlüssel des Kunden gebildet. Damit ist gewährleistet, dass niemand irgendwelche Features durch das Erraten von Dateinamen vom Server herunterladen kann.

- **Zeile 17:** Die URL zu dem kodierten Feature wird als Ergebnis des Web-Services zurückgeliefert.

6.2.4 Der Game-Feature-Shop Lite

Aufgabe war es, den Archiv-Builder aus dem Game-Feature-Shop herauszulösen. Das haben wir bereits durch den Web-Service erledigt. Nun bleibt noch die Implementation des neuen *Game-Feature-Shops Lite* und die Einbindung des im letzten Abschnitt erstellten Web-Services. Dies wird exemplarisch in der Programmiersprache PHP erfolgen. Der vollständige Code befindet sich auf der CD im Verzeichnis *shop*.

Wird der Benutzer zur Freischaltung eines Spieles aufgefordert, so wird eine fest kodierte oder konfigurierbare URL zu dem *Game-Feature-Shop Lite* aufgerufen. Diese Seite gilt als Einstiegsseite für den Shop und stellt eine ganz normale Webseite dar. Das Spiel übermittelt dabei die in *Tabelle 6.3* aufgeführten Parameter.

Name	Bedeutung
action	Gibt an, was getan werden soll. Mögliche Werte sind „download“, „acknowledge“ und „logout“. Die hier aufgeführten Parameter beziehen sich auf *action=download*.
gameid gameversion	Mit Hilfe von *gameid* und *gameversion* kann der Server die entsprechenden Features zur Auswahl stellen, die zur Spieleversion passen.
number	Hiermit kann ein Feature gekennzeichnet werden, das zum Weiterspielen benötigt wird. Dieses wird dann schon im Shop zum Kauf/Download vorausgewählt (0 = kein Feature, 1 = Freischaltung, ...)
affiliateid	PartnerID (kann vom Shop zu Abrechnungszwecken ausgewertet werden)
publickey0	Base64-codierter Public Key Modulus; mit diesem Schlüssel muss das Archiv verschlüsselt werden
publickey1	Base64-codierter Public Key Modulus; mit diesem Schlüssel sind die Postingparameter signiert
signature	Base64-codierte Signatur über die gesendeten Parameter
crc	CRC32 Summe über alle Postingparameter (inkl. Signatur)
archiveOk	Wird nur bei *action=acknowledge* übertragen und gibt den Fehlercode an. Ein Wert von 0 bedeutet, dass kein Fehler aufgetreten ist. Wurde das Archiv fehlerhaft heruntergeladen wird eine 1 gesendet. Konnte das Archiv nicht installiert werden, wird eine 2 gesendet.

Tabelle 6.3: *An den Game-Feature-Shop übermittelte Parameter*

Zuerst muss die CRC32-Checksumme der übertragenen Parameter überprüft werden. In einem zweiten Schritt folgt der Test der Signatur, um sicherzustellen, dass die Parameter nicht (beabsichtigt oder unbeabsichtigt) geändert wurden. Beides geschieht für den Benutzer unsichtbar im Hintergrund.

Da mit dem Herunterladen der Kauf eines Features und damit eine Geldtransaktion verbunden ist, muss sich der Benutzer erst einmal einloggen. Dazu kann er einen bereits früher erstellten Account verwenden oder einen neuen anlegen. Ist dies erfolgreich, so wird die Anfrage an den Web-Service geschickt. *Listing* 6.8 zeigt den dafür zuständigen Quellcode.

```
require_once('SOAP/Client.php');
$wsdl = new SOAP_WSDL('http://localhost:8080/axis/services/Archive?wsdl');
$Web-Service = $wsdl->getProxy();
$result = $Web-Service->createArchive((int)$gameid, $originalname, $featureurl,
    $publickey0);
```

Listing 6.8: *Aufruf des Web-Service via PHP*

Dank der SOAP Client- und Serverklassen des PEAR Projektes[1] gestaltet sich dies als leichtes Unterfangen. Zuerst muss die Basisklasse für den SOAP-Client eingebunden werden. Mit Hilfe des *SOAP_WSDL*-Objektes wird das WSDL-File des Web-Services, welches man über die bereits erläuterte URI erhält, analysiert. `$wsdl->getProxy()` liefert eine Instanz des Web-Services, die ganz normal aufgerufen werden kann, als wäre sie ein lokales Objekt. In der vierten und letzten Zeile werden die benötigten Parameter an den Web-Service übergeben. Die URL zum Featuredownload (oder ein eventuelles Fehlerobjekt) befindet sich in `$result`.

Bestätigt der Kunde seine Auswahl, wird mittels eines ActiveX-Controls der Download des Features automatisch gestartet. Der Client quittiert den Empfang der Datei mittels einer *Acknowledge*-Meldung (*action=acknowledge*) an den Server und schließt das Browser-Control mit einer weiteren Nachricht (*action=logout*). Der Client ist nun freigeschaltet und das Spiel kann fortgesetzt werden.

6.2.5 Offene Punkte

Im Rahmen der Diplomarbeit blieben einige Fragen unbeantwortet, die hier nicht verschwiegen werden sollen.

Um die Implementation des Prototyps so kurz und übersichtlich wie möglich zu halten, wurde auf eine Datenbank und die damit verbundene Zugriffsschicht zur persistenten Datenaufbewahrung verzichtet. An dessen Stelle traten einfache Textdateien, die die benötigten Informationen wie herkömmliche Konfigurationsdateien in einer flachen Struktur speichern. Diese Lösung bietet sich nicht für das Endprodukt an. Verwaltung, Pflege und Backup großer Datenmengen sollten in der Regel in einer Datenbank erfolgen. Dies war aber für die Beweisführung der Funktionstüchtigkeit nicht notwendig.

Zum Testen des Web-Services stand mir ein AMD Duron 900MHz mit 256MB Hauptspeicher zur Verfügung. Für eine Applikation wie Tomcat deutlich unterdimensioniert. So ist es auch nicht weiter verwunderlich, dass immer der erste Request nach dem Start des Servers sowie nach einer längeren Ruhezeit in einem Timeout endete. Tomcat agiert als Servlet-Container für AXIS und somit für unseren Web-Service. Dieser muss nun bei einer Anforderung in den Cache geladen werden, was auf meiner Testmaschine mitunter zu längeren Wartezeiten und dem damit verbundenen Timeout führte. Ein hinreichend ausgestatteter Computer sollte das Problem allerdings lösen.

Auch in der eigentlichen Implementation des *Game-Feature-Shops Lite* hat sich eine geringfügige Schwierigkeit eingeschlichen. Die Konzeption sieht eine Erstellung einer Signatur über den Request-String, der an den Shop geschickt werden soll, vor. Dieser wiederum muss die Signatur mit dem ebenfalls mitgeschickten öffentlichen Schlüssel verifizieren. PHP bringt hier allerdings unzureichende Unterstützung mit. Zwar wird mittels der OpenSSL-

[1] `http://pear.php.net/`

Bibliotheken der RSA Algorithmus in PHP zugänglich gemacht, diese fordern aber Zertifikate als Eingabe, die uns nicht zur Verfügung stehen. Zur Lösung des Problems wurde ein weiterer Web-Service vorgeschlagen. Dieser soll eine übermittelte Signatur einer Zeichenkette überprüfen und die Gültigkeit als Ergebnis zurück liefern. Hier treten aber wieder zwei neue Probleme auf. Da jede Anfrage an den Server laut Design einen solchen Check enthalten soll, muss mit deutlich erhöhtem Traffic gerechnet werden. Andererseits wurde die Signaturüberprüfung nur eingeführt, weil der Quelle (dem Client) oder dem Übertragungsweg dazwischen nicht vertraut wird. Nun soll aber dieser Check wieder über ein Netzwerk ausgelagert werden. Frage: Kann ich der Antwort des Web-Services vertrauen? Da der RSA-Algorithmus weithin bekannt ist und die Formeln zur Berechnung offengelegt sind, schlage ich eine einfache Umsetzung in PHP vor. Somit fällt der Netzwerkverkehr weg und die Signatur wird direkt vom *Game-Feature-Shop Lite* verifiziert. Die dabei erstellten Funktionen könnten in einer Bibliothek ausgelagert werden, die der Öffentlichkeit zur Nutzung zur Verfügung gestellt werden können.

6.3 Test

Zum Test des finalen Programms wurden drei Rechner benutzt – ein Rechner mit dem Client, einer für den Web-Service und ein dritter für den *Game-Feature-Shop Lite*. Dies sollte den später ebenso verteilten Rechnerverbund darstellen und etwaige Probleme in der Netzwerkkommunikation zwischen den Komponenten frühzeitig aufzeigen.

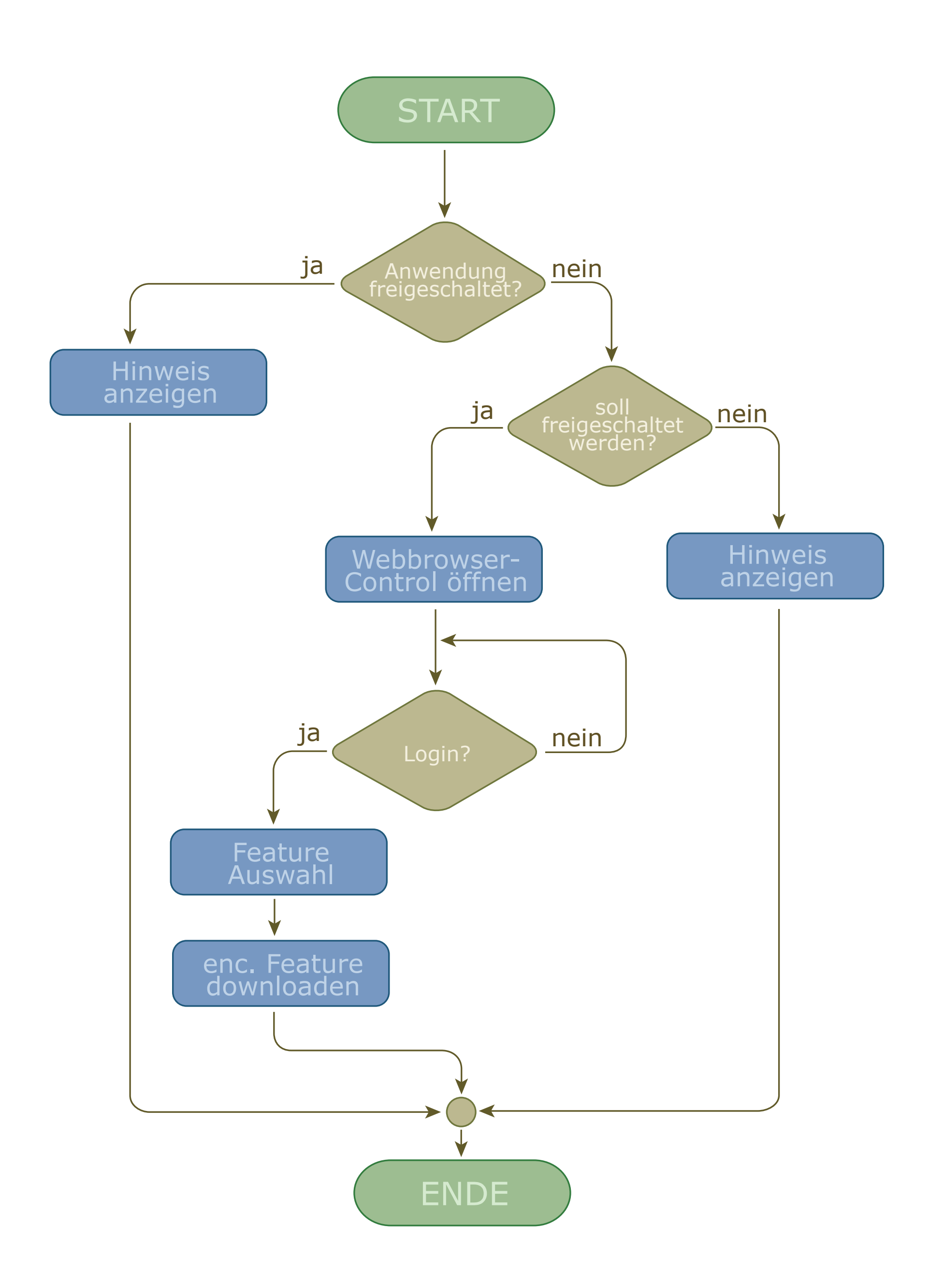


Abbildung 6.1: Programmablaufplan der Testapplikation

Da ich nur die Server-Komponenten entwickelt habe und der Client nicht Gegenstand dieser Arbeit war, half mir Stefan Richter (ein Mitarbeiter der 4FriendsOnly.com Internet

Technologies AG) mit einer Testapplikation aus. Die Anwendung besteht lediglich aus einer Routine zur Überprüfung des Zustandes der Freischaltung und den notwendigen Funktionen zur Freischaltung selber. *Abbildung 6.1* zeigt den Programmablaufplan.

Alle für die Nachbildung des Setups notwendigen Programme und Skripte befinden sich auf der beigefügten CD. Es bedarf einiger Anpassungsarbeit, um die hier aufgezeigten Komponenten bei sich zum Laufen zu bekommen. Hauptsächlich geht dies mit der Einstellung der URLs und IPs für die Rechner einher. Für den Web-Service gilt folgendes:

- Anpassen der Datei `Config.java` in dem Verzeichnis *java/src/com/ffo/gfp/webservices/ArchiveBuilder*. Hier ist einmal der Pfad zu der Konfigurationsdatei sowie der Pfad zum Abspeichern der kodierten Features einzustellen.
- Ändern der URI des Web-Services in der Datei `ArchiveServiceLocator.java` Zeile 13. Das wäre auch durch Ändern der URI in dem WSDL-File und Neugenerierung der Klassen möglich.

Nachdem die Java-Klassen mit dem *javac*-Compiler übersetzt wurden, kann das „Deployment“ nach Anleitung erfolgen[2]. Für den Shop gibt es lediglich eine Sache, die geändert werden muss:

- In der Zeile 52 der Datei `download.php` im Verzeichnis *shop/action* ist der Hostname und der Pfad zu den Features anzupassen. Diese URL wird dem Web-Service übergeben.

Vorausgesetzt die PHP-Engine sowie das SOAP-Paket des PEAR-Projektes sind installiert, reicht es, die Dateien des Shops in das Webroot zu kopieren. Als letztes muss dem Testclient noch gesagt werden, auf welchem Rechner er den Shop finden kann. Dazu muss erst einmal die Datei `Register.bat` im Verzeichnis *test_applikation* gestartet werden, damit die 4FO-DLL registriert werden kann. Um das Programm zu konfigurieren, ist folgendes zu tun:

- Nach dem ersten Start des Testclients wird eine Konfigurationsdatei in *C:/Programme/EasyGFP/203.ini* angelegt. In der [Global] Sektion ist die Zeile: *ServerURL=http://game.feature.shop.de/path/to/shop* hinzuzufügen.

[2] `http://ws.apache.org/axis/java/user-guide.html`

Kapitel 7

Zusammenfassung und Ausblick

Die vorliegende Arbeit hatte die Aufgabe, einen fairen Software-Kopierschutz basierend auf einer Client/Server-Architektur weiter zu entwickeln. Das Ergebnis ist die *Game-Feature-Plattform Lite*.

Zu Beginn der Arbeit wurde eine Einführung in das Urheberrecht gegeben. Wir haben gelernt, welchen Zweck der Gesetzgeber mit dessen Einführung verfolgte. Doch wie alles andere im Leben gibt es auch hier eine Schattenseite. Ist es in seiner aktuellen Version fair? Für die Industrie sicherlich. Klar ist, dass jede der Parteien seine Rechte gesichert wissen möchte. Der Rechteinhaber steht zweifelsohne gut da. Das Gesetz gibt ihm weitreichende Befugnisse gegen Urheberrechtsverletzer vorzugehen. Doch ist das wirklich förderlich oder beschränkt es eher die kommende Entwicklung? Es ist schwer eine objektive Wertung abzugeben. Das sogar die Privatkopie eines rechtmäßig erworbenen Spieles oder einer CD der Industrie ein Dorn im Auge ist, lässt sich aus der Sicht des Ottonormalverbrauchers nur schwer nachvollziehen. Auch sind Digital-Rights-Management-Systeme (DRM) beim „gemeinen Volk" wenig beliebt, da sie dem Anwender die Kontrolle zu entziehen versuchen. Die Zukunft ist ziemlich ungewiss. Für die Industrie und deren Lobbyisten ist die Richtung klar. Anstatt zwei Milliarden Gewinn sollen es nächstes Quartal eben drei Milliarden sein. Doch wer soll das bezahlen? Der Gesetzgeber täte wohl gut daran, weniger auf die Forderungen der Lobbyisten zu hören und seine Aufmerksamkeit Projekten wie die der Anti-DMCA-Organisation[1] zu zuwenden. Jedoch gibt es täglich nach wie vor Millionen von Urheberrechtsverletzungen in den Tauschbörsen. Die Recording Industry Association of America (RIAA) muss wohl festgestellt haben, dass es zu aufwendig ist, jedes Vergehen einzeln zu ahnden. Die logische Schlussfolgerung wäre doch, Programme wie eMule[2] und BitTorrent[3] gleich zu verbieten. So sieht eine neue Gesetzesvorlage, die da *Inducing Infringement of Copyrights Act of 2004* (Vgl. [PUB04]) heißt, vor, dass das vorsätzliche Verleiten zu einer Urheberrechtsverletzung bereits strafbar ist. Sollte das Gesetz angenommen werden, macht sich nicht nur der strafbar,

[1] http://www.anti-dmca.org/
[2] http://www.emule-project.net/
[3] http://bitconjurer.org/BitTorrent/

der die Verletzung begeht, sondern auch derjenige, der ihn dazu verleitet hat. Das würde das Ende einer jeden Tauschbörse bedeuten – vermutlich auch des uns bekannten Internets. Wer kann dann noch ruhigen Gewissens urheberrechtlich geschütztes Material auf einer Web-Seite anbieten? Er muss ja befürchten, dass ein anderer dieses herunterladen könnte.

Im Anschluss daran wurden verschiedene Vertriebsformen für Software vorgestellt. Wir haben uns dabei auf Low-Budget-Programme beschränkt. Die allseits bekannte Shareware hat hier den höchsten Stellenwert. Das Try-And-Buy-Prinzip erfreut sich größter Beliebtheit. Jedoch sind Systeme, die das Play-On-Demand (ähnlich dem Video-On-Demand) Prinzip umsetzen, stark im Kommen. Ist es nicht so, dass wenn man eine Datei per E-Mail erhält, aber das zugehörige Programm zum Öffnen fehlt, man nicht gleich hunderte von Euro für eine Lizenz des gesamten Softwarepaketes ausgeben möchte? Wie einfach wäre es doch, auf die Seite des Herstellers im Internet zu gehen und eine Nutzungslizenz für einen Tag zu erwerben. Auch das Potential von Freeware und damit verbundener Open-Source ist nicht zu unterschätzen. Entgegen der allgemeinen Vorstellung kann man auch hiermit Geld verdienen. Sicherlich, das eigentliche Programm lässt sich zumeist kostenlos aus dem Internet (sehen wir von den Kosten für die Internetverbindung selber einmal ab) herunterladen. Doch ein bereits praktiziertes Geschäftsgebahren ist es, das selbst entwickelte Programm zum freien Download anzubieten. Dokumentation, Support und Kundentraining unterliegen einem Obolos.

Das erste Kapitel endete mit dem Einblick in zwei von Microsoft entwickelte Systeme. Die *Next-Generation Secure Computer Base* sollte der Versuch einer sicheren Umgebung für Software werden, die nicht durch Hacker oder Viren verändert werden kann, Software, die ihren eigenen abgespeicherten Daten vertrauen kann. In der zukünftigen Windows-Version sollten alte wie neue Programme nebeneinander koexistieren. Das war zumindestens der Plan. Doch noch während diese Arbeit entstand, warf Microsoft die ganzen Vorstellungen und Ideale über den Haufen. Auf der diesjährigen *WinHec 2004* gab Microsoft ein Update zu der NGSCB Initiative bekannt. Vom streng getrennten zweigeteilten Windows mit unsicherer „Left-Hand-Side“ und NGSCB-gestählter „Right-Hand-Side“ war nichts mehr zu sehen. Statt einer streng isolierten rechten Seite soll es nun isolierte Compartments („Bereiche“) geben. Grund für die Veränderungen sei das Feedback von Kunden – gemeint sind dabei wohl Soft- und Hardware-Hersteller, keine Endanwender. Offenbar waren nicht allzuviele Entwickler bereit, zur Nutzung der Sicherheitsfunktionen spezielle Agenten und sichere Brückenprotokolle schreiben zu müssen. Tot ist NGSCB damit keineswegs, doch mutiert das ursprüngliche Konzept offenbar fleißig vor sich hin.

Das andere System ist die weiterhin strittige Produktaktivierung von Microsoft. Anvisiertes Ziel sollte die Eindämmung von Raubkopien sein. Doch wie so oft hat auch hier der gute und brave Bürger das Nachsehen. Er wird zu einer Aktivierung gezwungen, die trotz zahlreicher Versicherungen von der Seite von Microsoft einen bitteren Nachgeschmack lässt. Stellen sie sich vor, ihr neu gekaufter Fernseher zeigt kein Bild mehr, bis sie ihn beim Hersteller

registriert haben. Er könnte ja aus einer Zweit- oder Drittproduktion aus Thailand kommen. Der versierte Internetbenutzer lädt sich einen Crack von einschlägig bekannten Seiten herunter und ist vor zukünftigen Querelen seines Betriebssystems oder Office-Paketes sicher. Doch der totale Ausschluss von Raubkopierern hat sich als gar nicht so vorteilhaft herausgestellt. Jedem dürfte bewusst sein, dass Microsofts Betriebssystem nur deshalb einen so hohen Verteilungs- und Bekanntheitsgrad hat, weil es in den Anfangszeiten stark kopiert wurde. Im gewissen Sinne profitiert Microsoft heute davon. Seit dem Service-Pack 1 von Windows XP war es vielen (illegalen) Benutzern nicht mehr möglich, Sicherheitsupdates einzuspielen. Die Folge: eine hohe Anzahl von verseuchten oder manipulierten Systemen. Die Presse schob natürlich Microsoft den schwarzen Peter zu – vielleicht nicht ganz zu Unrecht.

Im zweiten Kapitel wurde ein kurzer Einblick in die Kryptografie gewährt. Die Ver- und Entschlüsselung von Daten ist ein integraler Bestandteil der Game-Feature-Plattform und deren Weiterentwicklung. Ziel war es, lediglich Grundbegriffe zu klären. Bei weiterführendem Interesse wird auf die einschlägige Literatur verwiesen.

Die Game-Feature-Plattform der Firma 4FriendsOnly.com AG war Bestandteil des vierten Kapitels. Man hatte bereits ein Konzept erarbeitet, dass die oben genannten Kritikpunkte anzugehen versucht. Die erste Version ist nie perfekt. So wurden in *Kapitel 4.3 (S.40)* Probleme besprochen, die durch die tägliche Nutzung aufgetreten sind.

Die Arbeit wurde mit der Konzeption und Umsetzung der *Game-Feature-Plattform Lite* abgeschlossen. Sie stellt eine Weiterentwicklung der existierenden Game-Feature-Plattform dar und soll deren Schwächen beseitigen. In wie weit dies gelungen ist, wird wohl nur die Praxis zeigen können. Abschließend kann man sagen, dass das Konzept der GFP durchaus Potential hat, ein exzellentes Vertriebssystem zu werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[VGW04] Verwertungsgesellschaft WORT: Geschichte des Urheberrechts und geistiges Eigentum, Stand: August 2004
http://www.vgwort.de/geschichte.php

[WIK04a] Wikipedia: Anwendungen und Gerichtsverfahren zum DMCA, 12. März 2004
http://de.wikipedia.org/wiki/DMCA

[KRE03] Stefan Krempl: Die rechtlich geschützte Kopierschutz-Gesellschaft, 17. April 2003
http://www.heise.de/tp/deutsch/special/copy/14624/1.html

[EUCD01] Richtlinie 2001/29/EG des Europäischen Parlaments, 22. Mai 2001
http://europa.eu.int/cgi-bin/eur-lex/udl.pl?REQUEST=Seek-Deliver&COLLECTION=oj&SERVICE=all&LANGUAGE=de&DOCID=2001l167p0010

[URHG] Bundesministerium der Justiz: Gesetz über Urheberrecht und verwandte Schutzrechte, Stand: August 2004
http://bundesrecht.juris.de/bundesrecht/urhg/gesamt.pdf

[GG] Bundesministerium der Justiz: Grundgesetz für die Bundesrepublik Deutschland, Stand: August 2004
http://bundesrecht.juris.de/bundesrecht/gg/gesamt.pdf

[WIK04b] Wikipedia: Public Domain, 22. August 2004
http://de.wikipedia.org/wiki/Public_domain

[WIK04c] Wikipedia: Freeware, 30. August 2004
http://de.wikipedia.org/wiki/Freeware

[HWF92] Hans Werner Fromme: Die Besten Deutschen Shareware Programme Band 2, Systhema Vlg., München, Februar 1992
ISBN: 389390347X

[KNO96] Jim Knopf: The Origin of Shareware, Stand: August 2004
http://www.freewarehof.org/sstory.html

[MS02] Microsoft Palladium: A Business Overview, Stand: August 2002
http://www.microsoft.com/PressPass/features/2002/jul02/0724palladiumwp.asp

[MS04a] Microsoft Palladium: Archives, Stand: August 2004
http://www.microsoft.com/resources/ngscb/archive.mspx

[MS04b] Microsoft: Produktaktivierung, 26. Mai 2004
http://www.microsoft.com/germany/ms/produktaktivierung/

[RSA00] RSA Laboratories: PKCS #10 v1.7: Certification Request Syntax Standard, 26. Mai 2000
ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf

[WOB98] Reinhard Wobst: Abenteuer Kryptologie. Methoden, Risiken und Nutzen der Datenverschlüsselung, Addison-Wesley, 1998
ISBN: 3827314135

[LAU99] Alexander Lauert: Elektronisches Bezahlen, 1999
http://ddi.cs.uni-potsdam.de/Lehre/e-commerce/elBez2-5/index.html

[WIK04d] Wikipedia: Symmetrisches Kryptosystem, 2. Juli 2004
http://de.wikipedia.org/wiki/Symmetrisches_Kryptosystem

[WIK04e] Wikipedia: Asymmetrisches Kryptosystem, 19. Juli 2004
http://de.wikipedia.org/wiki/Asymmetrisches_Kryptosystem

[MOV03] Handbook of Applied Cryptography, 5. Auflage; Alfred J. Menezes, Paul C. van Oorschot und Scott A. Vanstone; 11. November 2003

[NUE02] Virtuelle Waren bezahlbar machen, Leipziger Informatik-Tage 2002, Leipzig, 26. und 27. September, 2002, in Von e-Learning bis e-Payment, Editor: K.P. Jantke, W.S. Wittig und J. Herrmann, ISBN: 3-89838-033-5, S. 356-364
http://www.4friendsonly.org/papers/LIT-2002-Endfassung-47.pdf

[NUE01] The Game Feature Platform, 46th International Scientific Colloquium, Ilmenau Technical University, September 24 - 27, 2001
http://www.4fo.de/download/gfp_full_iwk.pdf

[OES01] Bernd Oestereich: Erfolgreich mit Objektorientierung, Oldenbourg Verlag, Februar 2001
ISBN: 3486255657

[OES01] Andrew S. Tanenbaum, Maarten van Steen: Verteilte Systeme, Pearson Studium, April 2003
ISBN: 3827370574

[PUB04] Public Knowledge: Resource Room for the Inducing Infringement of Copyrights Act of 2004 (formerly known as the INDUCE Act), August 2004
http://www.publicknowledge.org/issues/induce-act/